



## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

|  |  |           |   |
|--|--|-----------|---|
| (51) International Patent Classification <sup>6</sup> :<br><b>G06F 15/42</b> |  | <b>A1</b> | (11) International Publication Number:<br><b>WO 99/35588</b>    |
|  |  |           | (43) International Publication Date:<br>15 July 1999 (15.07.99) |

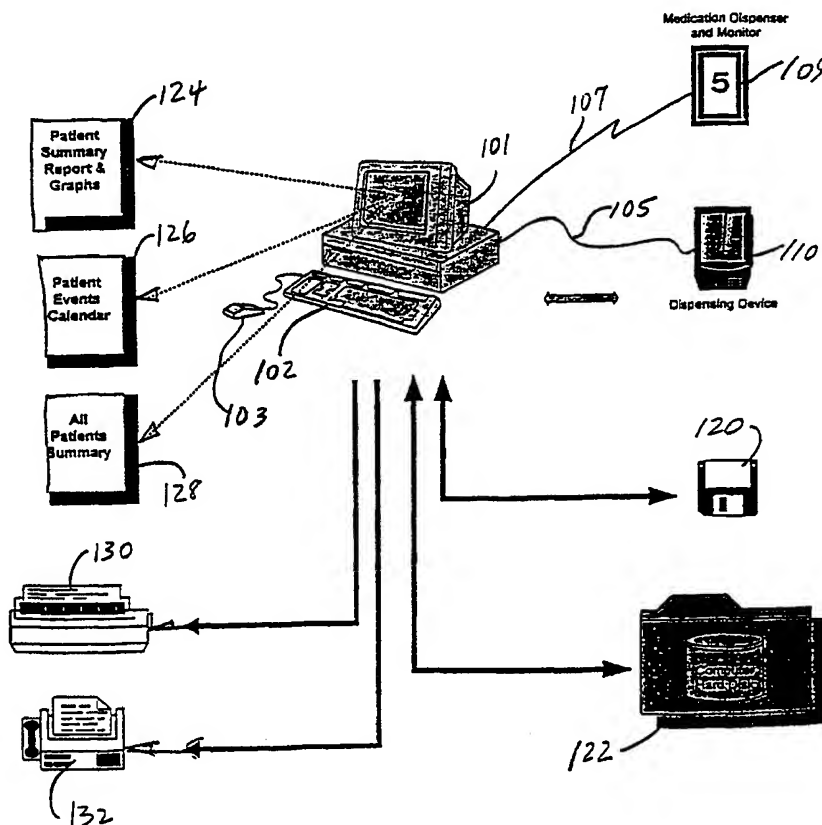
  

|   |  |
|---|--|
| <p>(21) International Application Number: PCT/US98/22830</p> <p>(22) International Filing Date: 29 October 1998 (29.10.98)</p> <p>(30) Priority Data:<br/> 60/071,107      12 January 1998 (12.01.98)      US<br/> 09/143,483      28 August 1998 (28.08.98)      US</p> <p>(71) Applicant (for all designated States except US): SANGSTAT MEDICAL CORPORATION [US/US]; 1505 Adams Drive, Menlo Park, CA 94025 (US).</p> <p>(72) Inventors; and<br/> (75) Inventors/Applicants (for US only): POULETTY, Philippe [FR/US]; 25 Oakhill Drive, Woodside, CA 94062 (US). HAMILTON, Richard, G. [US/US]; 4670 Amiens Avenue, Fremont, CA 94555 (US). ROSSI, Stephen, J. [US/US]; 655 Kansas Street #301, San Francisco, CA 94107 (US). McENROE, Debra, L. [US/US]; 566 Cutwater Lane, Foster City, CA 94404 (US).</p> <p>(74) Agents: CARLSON, Stephen, C. et al.; McDermott, Will &amp; Emery, Suite 300, 99 Canal Center Plaza, Alexandria, VA 22314 (US).</p> | <p>(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).</p> <p><b>Published</b><br/> <i>With international search report.</i><br/> <i>Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p> |
|---|--|

(54) Title: SYSTEM AND METHOD FOR MANAGING ADMINISTRATION OF MEDICINE

## (57) Abstract

A method for managing doses of medication delivered to a patient is described. A computer system (101) receives dosage data and administration data that represent, respectively, times and quantities for taking a drug that are prescribed for a patient, and the times and quantities the drug is delivered to the patient. Based on the dosage and administration data, compliance information is generated and displayed, representing the degree to which a drug has been delivered in accordance with the dosage data. In one aspect, a calendar (126) in the form of a grid comprised of grid elements is displayed. Each grid element represents a period, such as a day in a month, and contains one or more icons. An icon's appearance indicates whether a particular dose was delivered properly, when a grid element is selected by a user, more detail is displayed about the administration of the drug for the respective day.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

|    |                          |    |                     |    |                       |    |                          |
|----|--------------------------|----|---------------------|----|-----------------------|----|--------------------------|
| AL | Albania                  | ES | Spain               | LS | Lesotho               | SI | Slovenia                 |
| AM | Armenia                  | FI | Finland             | LT | Lithuania             | SK | Slovakia                 |
| AT | Austria                  | FR | France              | LU | Luxembourg            | SN | Senegal                  |
| AU | Australia                | GA | Gabon               | LV | Latvia                | SZ | Swaziland                |
| AZ | Azerbaijan               | GB | United Kingdom      | MC | Monaco                | TD | Chad                     |
| BA | Bosnia and Herzegovina   | GE | Georgia             | MD | Republic of Moldova   | TG | Togo                     |
| BB | Barbados                 | GH | Ghana               | MG | Madagascar            | TJ | Tajikistan               |
| BE | Belgium                  | GN | Guinea              | MK | The former Yugoslav   | TM | Turkmenistan             |
| BF | Burkina Faso             | GR | Greece              |    | Republic of Macedonia | TR | Turkey                   |
| BG | Bulgaria                 | HU | Hungary             | ML | Mali                  | TT | Trinidad and Tobago      |
| BJ | Benin                    | IE | Ireland             | MN | Mongolia              | UA | Ukraine                  |
| BR | Brazil                   | IL | Israel              | MR | Mauritania            | UG | Uganda                   |
| BY | Belarus                  | IS | Iceland             | MW | Malawi                | US | United States of America |
| CA | Canada                   | IT | Italy               | MX | Mexico                | UZ | Uzbekistan               |
| CF | Central African Republic | JP | Japan               | NE | Niger                 | VN | Viet Nam                 |
| CG | Congo                    | KE | Kenya               | NL | Netherlands           | YU | Yugoslavia               |
| CH | Switzerland              | KG | Kyrgyzstan          | NO | Norway                | ZW | Zimbabwe                 |
| CI | Côte d'Ivoire            | KP | Democratic People's | NZ | New Zealand           |    |                          |
| CM | Cameroon                 |    | Republic of Korea   | PL | Poland                |    |                          |
| CN | China                    | KR | Republic of Korea   | PT | Portugal              |    |                          |
| CU | Cuba                     | KZ | Kazakhstan          | RO | Romania               |    |                          |
| CZ | Czech Republic           | LC | Saint Lucia         | RU | Russian Federation    |    |                          |
| DE | Germany                  | LI | Liechtenstein       | SD | Sudan                 |    |                          |
| DK | Denmark                  | LK | Sri Lanka           | SE | Sweden                |    |                          |
| EE | Estonia                  | LR | Liberia             | SG | Singapore             |    |                          |

SYSTEM AND METHOD FOR MANAGING ADMINISTRATION OF MEDICINE  
RELATED APPLICATIONS

## SYSTEM AND METHOD FOR MANAGING ADMINISTRATION OF MEDICINE RELATED APPLICATIONS

This application claims priority from prior U.S. provisional application serial number 60/071,107 filed on January 12, 1998, entitled "Method and System for Monitoring Doses,"  
5 which is incorporated by reference in its entirety as if fully set forth herein.

### COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction in paper copies by anyone of the patent document or the patent disclosure, as it appears in the  
10 Patent & Trademark Office patent file or records, but otherwise reserves all other copyright rights whatsoever.

### FIELD OF THE INVENTION

The present invention relates generally to computer systems. The invention relates more specifically to managing administration of medicine, monitoring dosages of drugs given  
15 to patients, and the like.

### BACKGROUND OF THE INVENTION

Monitoring dosages of drugs or medicines for patients requires communication among several levels. First, a physician must diagnose and prescribe a dosage for a patient. The medication must then be distributed accurately and, finally, the patient or a care provider must  
20 ensure that the dosages are properly administered to or taken by the patient.

For many reasons, ensuring that accurate dosages are delivered to a patient in a consistently timely manner can be difficult despite the importance of accurate administration in many instances.

Therefore, it is desirable to provide a method of automating the delivery of medicine  
25 and monitoring the delivery of medicine.

Moreover, special challenges are presented in managing patients who are taking more than one medication. Elderly patients on multiple medications may have difficulty keeping track of whether they have taken all their medications, when, and in what quantity. In the clinical setting, proper administration of multiple medications to acutely ill patients is  
30 challenging for care providers.

Thus there is a need to track multiple medications and multiple dispensing mechanisms, and to present data for all such dispensers in a report.

To facilitate the proper administration of medication and the tracking of when it is administered, medication dispensing devices are used. Conventional medication dispensing  
35 devices typically include a medicine container and an alarm mechanism which notifies a patient at the time intervals the dose(s) are due. Each time the patient opens the container, the device records the event and the time it occurred. One example of a conventional medication dispensing device is a jar with lid which incorporates an alarm mechanism and a recording mechanism. When the lid is removed, the recording mechanism records this event and the  
40 time it occurred.



One drawback to conventional dispensing devices is that they do not control access to medicine or the quantities dispensed. Thus, there is little assurance that when a dispensing device is opened, the proper amount is dispensed. Another drawback is that once opened, the dispensing devices may be re-opened immediately. Thus a confused elderly patient, having  
5 forgotten the dose they just took, may take another far too soon.

Thus, there is further need for a system that controls dispensing times and amounts and which tracks those times and amounts.

#### SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent from the  
10 following discussion, are fulfilled by the present invention, which comprises, in one aspect, a method for managing doses of medication delivered to a patient. Generally, a computer system receives dosage data and administration data. The dosage data represents a drug prescription, and includes, but is not limited to, one or more times for taking the drug, the quantities in which the drug is to be taken by the patient, or a combination thereof. The administration data  
15 represents when and in what quantities each dose in a set of doses of the drug is actually delivered to the patient. Based on the dosage and administration data, compliance information is generated and displayed. Compliance information indicates the degree to which a drug has been delivered in accordance with the dosage data. The compliance information can be displayed in variety of forms.

According to another aspect, a calendar in the form of a grid comprised of grid  
20 elements is displayed. Each grid element represents a period, such as a day in a month, and contains one or more icons. An icon's appearance indicates whether a particular dose was delivered properly. For example, a green square icon indicates that a dose was delivered on time, and a triangular red icon indicates that a dose was not delivered. When a user selects a  
25 grid element, more detail is displayed about the delivery of the drug for the respective day. In particular, a graphical object is displayed that contains one or more icons for each dose delivered in the day. An icon's position along an axis of the graphical object reflects when a dose was delivered.

According to another aspect, data is generated that specifies what portion of a set of  
30 doses was delivered properly. The data includes values that indicate what portions of the doses were delivered, and what proportion of doses were delivered on time.

According to another aspect, dosage data is transmitted to a dosage-dispensing device. The dosage data includes times and quantities to deliver a drug to a patient. In addition, data representing a lockout period may be transmitted. The dosage-dispensing device dispenses the  
35 drug to the patient in accordance with the data transmitted to it.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

40 FIG. 1 is a block diagram illustrating a system for monitoring patient dosages.

FIG. 2 is a flow chart illustrating steps for a computer-implemented method for monitoring patient dosages.

FIG. 3 is a flow chart showing steps for retrieving data that is used in a system for monitoring the administration of doses to a patient.

5 FIG. 4 is flow chart showing steps for transmitting dosage information to a dosage-dispensing device.

FIG. 5A is block diagram depicting a calendar in the form of a grid.

FIG. 5B is a block diagram depicting a grid element and icons used to indicate patient compliance.

10 FIG. 5C is a block diagram depicting a graphical object used to graphically represent when doses were delivered.

Fig. 6 is a block diagram depicting a histogram showing dosage scores over period of time.

#### DETAILED DESCRIPTION

##### 15 OVERVIEW

One embodiment is a system and method for substantially automating the administration of patient dosages, the monitoring of the delivery of doses, whether or not timely and whether or not accurate in amount, and the accumulation of data for individual patients representing administration data over an extended period of time.

20 Another embodiment encompasses accumulation of data for each patient from a plurality of dosage dispensing devices, and the assimilation of such data into reports which may be either specific for the particular patient, or an accumulation of data for an entire range of patients. In this way, more accurate dispensing of doses is achieved, as well as more accurate monitoring to facilitate detection of whether prescribed doses are being properly  
25 administered to the patients.

A preferred embodiment provides a computer-implemented method for monitoring patient dosages by retrieving administration data, including times and amounts of medication prescribed for a patient, retrieving patient data, including times and amounts of medication delivered for the patient, determining evaluation data by analyzing the retrieved dosing and  
30 patient data to determine compliance of the delivered medication to the prescribed medication, and displaying the evaluation data.

The method may include one or more of the following features. Patient data, including administration data, may be received from an associated device over a communications line, from local memory, or from user input. The data may be accumulated to provide a basis for  
35 patient evaluation. The patient data may be transmitted to a dosage-dispensing device, which dispenses doses to the patient in accordance with the received patient data.

The evaluation data may be displayed in a variety of ways, including display in a patient administration report that may indicate compliance of the delivered doses to the prescribed dosages. In one implementation, the data retrieved may be viewed in a scrollable  
40 tabular grid, with displayed values for all medication events, and dates, times and dose sizes

dispensed from the dosage dispensing device. In addition, non-medication events may be displayed, including "bottle replaced" or other ancillary but relevant data.

Additionally, evaluation data may be displayed in the form of a patient summary report which may, for example, include all information for a particular patient including name, ID, monitoring dates, drug, brand, and so on. In addition, a histogram may be prepared summarizing the patient's compliance, including calculation of a "compliance index" or similar quantification of the patient's overall compliance with the prescribed dosing plan. The evaluation data may be displayed for varying periods, such as a week, a month, or a shorter or longer period, and may be displayed in graphical form including options for displaying doses delivered, missed, or delivered but not within compliance parameters. The data may also be displayed in calendar form.

In many instances, patients undergoing treatment may have multiple dosage dispensers. In a manner similar to the single dispenser arrangement discussed above, data for each such dispenser can be tracked and presented in a merged patient summary report.

Likewise, a summary of all patients may be provided which may provide, in either graphical or tabular form, any of the selected data including name, ID, compliance index, dosage, time of day, or any other field. Histograms may also be developed across the patient class.

Evaluation data may be provided in any suitable format, such as a data file or hard copy. For example, the data may be printed or transmitted to a remote facsimile machine.

According to one embodiment, the delivery of doses of multiple patients is monitored. In this embodiment, a preferred method comprises retrieving dosage data, including times and amounts of medication prescribed for a plurality of patients, retrieving patient data, including times and amounts of medication delivered for the plurality of patients, determining evaluation data by analyzing the retrieved dosing and patient data for the plurality of patients to determine overall compliance of the delivered medication to the prescribed medications, and displaying the evaluation data.

Another embodiment includes a memory device storing computer readable instructions for aiding a computer to implement a method for monitoring patient dosages such as that described above.

Yet another embodiment provides a system for monitoring patient dosages including a computer implementing a method such as that described above.

#### MEDICINE ADMINISTRATION MANAGEMENT SYSTEM

Embodiments of the invention may be implemented on special purpose electronic or data processing hardware, software applications running on general purpose hardware, or a combination of both. For example, an embodiment may be implemented in a dose administration system that includes a computer system running one or more application programs that provide functions for manipulating dosing and patient data, having access through appropriate communications links to remote devices.

FIG. 1 shows an illustrative system incorporating the present invention, including personal computer 101 running application software. Computer 101 has access to both dosage

data and patient data. For example, as shown in FIG. 1, the computer 101 includes a communications link 105 that couples computer 101 to dosage dispensing device 110. The dosage dispensing device 110 may be, for example, the portable medication administration device described in U.S. Patent Application Serial No. 08/ 867,010 entitled Liquid Medication Dispenser Apparatus, filed on June 2, 1997 and naming as inventors Debra L. McEnroe, Robert A. Britts, Phillippe Pouletty and Ralph Levy, the entire contents of which are hereby incorporated by reference as if fully set forth herein. Dosage dispensing device 110 may be used to dispense, for example, an analgesic drug, opiate agonist or antagonist drug, or a immunosuppressive drug, such azathioprine, Tacrolimus, Sirolimus, mycophenolate, mofetil, and their chemical derivatives.

A portable medication administration device is a device which may be transported with the patient outside a medical facility such as a hospital or doctor's office, and which delivers multiple doses to the patient without immediate supervision by a registered medical clinician. Such dispensers are typically used by, for example, physicians and pharmacists, to input dosage data.

Communications link 105 enables the dosage data to be recorded at locations remote from the monitoring system, such as at medical facilities where medications are prescribed.

In the illustrated monitoring system, the computer 101 retrieves information relating to the patient data from data stored on diskette 120 or in a mass storage device, such as the computer's hard disk drive 122. This data typically includes a record of doses delivered to the patient and is typically created by the patient or a caretaker. As with the dosage information, this information may be input at remote locations, such as at a patient's home or a location where the medication is administered.

Of course, dosage and patient data may also be provided by alternative methods. For example, the data may be input directly by a user through the computer keyboard 102. The computer 101 can save input and retrieve information by downloading to the diskette 120 or hard drive 122, or if appropriate, may initiate to medication dispenser and monitor 109 a communications link 107. Communications link 107 may use electrical, electromagnetic, optical signals, or other signals that may carry digital data. These signals are exemplary forms of carrier waves transporting information.

Application software running on the computer 101 processes the dosage and patient data to determine monitoring information for patients. The monitoring information is provided to a user in the format of, for example, patient summary reports and graphs 124, event calendars 126, and summaries of groups of patients 128. The monitoring information can also be provided in hard copy via printer 130 or fax 132 through appropriate communication links.

Computer 110 may transmit data to dosage dispensing device 110 via communications link 105. The data may include times and quantities to administer a drug to a patient, and a value representing a lockout period. Dosage dispensing device 110 delivers a drug in accordance with the received data.

In one embodiment, computer 101 is a personal computer having an Intel or AMD-type processor and running the Microsoft® Windows 95 or Windows NT operating system, and equipped with volatile memory such as RAM and non-volatile memory such as a hard disk. A display device such as a CRT also forms part of computer 101.

#### MONITORING ADMINISTRATION OF MEDICINE TO A PATIENT

FIG. 2 is a diagram of a method of monitoring the administration of medicine to a patient. In one embodiment, the method of FIG. 2 is implemented in one or more application programs that are executed by computer 101.

At block 202, a computer such as personal computer 101 of the system of FIG. 1, begins execution of the application software. As shown in block 210, computer 101 retrieves dosage and patient data for a patient from stored data. As indicated by block 212, the steps of block 210 may involve retrieving previously stored data files from a mass storage device such as disk drive 122.

Alternatively, computer 101 may establish an appropriate communications link, such as a modem or ISDN line, to retrieve data from a remote device, such as the portable medication administration device illustrated in FIG. 1 and described in the above-referenced U.S. Patent Application Serial No. 867,010, filed June 2, 1997 and entitled Liquid Medication Dispenser Apparatus, previously incorporated by reference. In this alternative case, as indicated in block 214, the dispensing device 110 is connected to the computer 101 and prepared for communication with the computer.

At block 220, dosage and medicine administration information for a patient is reviewed. Specifically, updated patient data is processed by the application software and displayed as requested by a user. The application software may be adapted to manipulate the dosage and patient information as needed. For example, the software may monitor the dosages delivered to patients by recording times and amounts of doses taken by a specified patient, as indicated by the retrieved patent data. With access also to the dosage information for that patient, the software may determine, for example, compliance of a patient's delivered doses with the prescribed doses, either for specified dose times or over a period of time.

Block 220 may involve generating one or more reports, as shown by block 224. For example, the method may be used to generate calendars showing the dosing events indicating, for example, the times of prescribed doses for specific patients and whether the patient complied with those doses. The method may also generate summary reports and graphs reflecting the progress of treatment for specific patients, incorporating, for example, test results. Additionally, the method may generate summary reports for groups of patients, such as groups of patients taking the same medication or groups of patients of a specific physician.

The analyzed results may be stored and may be provided to a user. For example, the method may display the results on a computer monitor. Alternatively, as indicated in block 222, the computer 101 may provide hard copies of reports by printing to a printer or transmitting the results to a remote facsimile machine.

Optionally, as shown by block 230, the data is saved after it is reviewed. As indicated by block 232, the data is saved to the mass storage device from which it was retrieved.

Alternatively, as indicated in block 234, computer 101 may clear the memory of an external device from which the data was received and save a new copy of the data, or modify

appropriate parameters of the external device. A pre-defined format is used. For example, data read from the device 110 may be saved as one or more comma-delimited ASCII files on disk 122. Use of such a format enables the data to be human-readable, and allows the data to be imported into commercial, off-the-shelf application programs such as spreadsheets or word processors.

In one embodiment, the data is saved with a validation code that is computed at the time the file is saved. Whenever a saved data file is reopened, the code will be used to test and guarantee the validity of the data against corruption of the data or intentional modification by any means outside of the program. In a preferred embodiment, a relational database system such as the Microsoft Access Jet Engine is used for storing and retrieving all data.

At block 240, the operational sequence is complete.

#### RETRIEVING PATIENT DATA - INCLUDING DOSES AND TIME DELIVERED

FIG. 3 illustrates an embodiment of a method of retrieving data. FIG. 3 illustrates substeps involved in block 210 of FIG. 2 in greater detail.

At block 304, the computer system receives a request to read device data. For example, block 304 may involve receiving a request to read "current patient data" that is stored in the dispensing device 110. The request may be generated in response to, for example, a user selecting a program menu option in a graphical user interface ("GUI").

As shown by block 320, the system determines whether dosage or patient data for the requested patient already exists and has not been saved since a prior retrieval operation. If patient data for the requested patient already exists in memory and has not been saved during a prior retrieval, then in block 324, the system displays a prompt message to the user. The prompt message enables the user to select (1) canceling the request to retrieve patient data from the device, or (2) saving the prior data before continuing with the process of retrieving current patient data from the dosage-dispensing device. If the user chooses to cancel the request to retrieve the current patient data, then execution ends. If the user chooses to save the already existing data, then control flows to block 328, where the data is saved in a user specified file. Block 328 may involve displaying a dialog box or prompt to the user that requests the user to enter a file name or pathname. Control then flows to block 330.

At block 330, the current patient data is retrieved from the dosage-dispensing device and stored in a temporary buffer. The temporary buffer may be, for example, a temporary disk file or a buffer area in memory. At block 334, the data is checked to determine whether any transmission or data errors occurred during transmission from the dosage-dispensing device. For example, an 8-bit checksum algorithm can be applied to data received from a dispensing device 110 to detect errors. Such checksums are conventionally included by the dispensing device 110 in data that it transmits to computer 101. If any errors are detected, then at block

338, a message to the user is displayed, informing the user that errors exist in the data, and execution ends. If no transmission errors are detected, then control flows to block 340.

As indicated by block 340, the disk or other storage device is checked to determine whether any prior patient data for the patient has been retrieved and stored. If previous data  
5 has been retrieved from the device, then control flows to block 344. In this case, as shown by block 344, data for the patient is updated by merging the current patient data with the prior data. The merged data is stored in memory. A message is displayed informing the user that the merge has occurred.

As shown by block 348, the current data is stored. Alternatively, the merged data is  
10 stored, if merged data was created at block 344. The user interface is updated to reflect the addition of current patient data.

At block 360, a device retrieval dialogue is displayed, which is data about the just retrieved patient data. Such data can include patient name, the drug(s), prescribed doses per day, and the administration times.

#### 15 TRANSMITTING DOSAGE DATA TO DOSAGE DISPENSING DEVICE

In one embodiment, computer 101 transmits dosage data to dosage dispensing device 110. The dosage data is used by dosage dispensing device 110 to control the dispensing of medicine. The dosage data may represent medicine to deliver, administration times, quantities, and a lockout period. A lockout period is a period of time that must elapse after dispensing a  
20 dose before another dose may be administered or delivered to the patient. The dosage data may specify medicines that include, for example, an analgesic drug, opiate agonist or antagonist drug, or a immunosuppressive drug. An example of a dosage dispensing device that receives data specifying administration times and quantities and a lock out period, and then which operates in accordance to such data, is the portable medication administration device,  
25 described in U.S. Patent Application Serial No. 867,010, filed June 2, 1997 and entitled Liquid Medication Dispenser Apparatus, previously incorporated by reference.

The ability to transmit data to a dosage device that dispenses medicine accordingly provides significant advantages. The amounts of medicine that are actually dispensed to the patient may be controlled, and premature administration of doses may be prevented.

30 FIG. 4 is a diagram of a method of collecting dosage data from a user, such as a physician or other clinician, and transmitting the dosage data to a dosage dispensing device.

As shown by block 410, a request is received from a user to enter dosage data. The request may be generated in response to a user selecting a program menu option in a GUI. As indicated by block 420, current dosage data for the patient is retrieved from stored data. At  
35 block 430, a data entry screen or dialog box is displayed, showing the current dosage data as the default data.

As indicated by block 440, dosage data is received from the user. The dosage data includes prescribed administration times and quantities and a lockout period. For example, the user enters the following information:

40 Number of Doses

Quantity and Unit  
Times for Each Dose  
Lock-out Period

As shown by block 450, the dosage data is transmitted to a dosage dispensing device,  
5 such as device 110 shown in FIG. 1. At block 460, the dosage data is stored in a mass storage  
device of a computer system, for example, hard disk 122 of computer 101.

In an embodiment of the present invention, the application software may be adapted to  
analyze additional data. This may include device monitoring data, such as the time a drug  
bottle was changed, temperature monitoring data, battery status, times data was downloaded  
10 from a dosage dispensing device, data identifying the bottle of the drug, such as data read  
from a bar code. Patient data may include test results measured at specified times to measure  
the effect of the administered dosages, or information on multiple drugs dispensed by a dosage  
dispensing device. Dosage data may include proper dosages of specified medications, as well  
as an indication of possible side effects and information regarding whether the dosage should  
15 be altered should those side effects be detected. In such a case, the application software may  
be adapted to provide an analysis of the effectiveness of the administered dosage.

#### EXEMPLARY GENERATION OF COMPLIANCE INFORMATION

To help determine whether a patient is administering a drug properly, compliance  
information is generated and displayed to a user. The system may display such compliance  
20 information in many forms. For example, the system may display a calendar that indicates  
whether particular doses were delivered properly. As another example, the system may display  
one or more compliance indexes, such as the percent of daily doses delivered or the percent of  
doses delivered on time. The compliance information may be generated by, for example, a  
computer system executing a computer program according to the source code set forth in the  
25 Appendix.

#### CALENDAR SHOWING PATIENT COMPLIANCE

FIG. 5A is a block diagram depicting a calendar 500. In the preferred embodiment, one  
or more calendars 500 are displayed to graphically convey user compliance information on a  
computer display, or other output device such as a printer.

30 Calendar 500 of FIG. 5A comprises a grid 502, which includes one or more grid  
elements 520. Each grid element 520 represents a particular day of the month, and may  
contain one or more icons 521 for each dose due on the particular day. The calendar 500 may  
also include a legend 523 that identifies each icon 521 with a descriptive label. Thus, each  
calendar 500 provides a snapshot display to the user of the dosages due for a particular patient  
35 throughout a particular month.

FIG. 5B shows grid element 520 in greater detail. Grid element 520 of FIG. 5B  
pertains to the second day of a particular month, as indicated by the numeric day value 540.  
Grid element 520 includes one or more icons 521 selected from among a new dosage icon  
522, wrong time icon 524, on-time icon 526, and missed dose icon 528. The particular icons



521 that appear in a particular grid element 520 depend upon the content of the data previously entered for the patient by the user.

New dosage icon 522 is displayed so that it reflects the day the dosage was changed, as specified by, for example, dosing data retrieved from a dosage dispensing device 110. The new dose size may be displayed within new dosage icon 522. For example, new dosage icon 522 may include text showing that the dosage is "250 mg".

Preferably, wrong time icon 524, and missed dose icon 528 each are displayed with different patterns that indicate whether a dose was delivered properly. For example, wrong time icon 524 is a square shaped icon that is displayed in a first color, such as brown or tan, and is displayed for a dose that was delivered at the wrong time. A dose is delivered at the wrong time if it was delivered to the patient at a time outside the scheduled administration time.

Similarly, on-time icon 526 may be a green colored icon, and is displayed for a dose that was delivered on time. A dose is delivered on time if it was delivered to the patient within the scheduled administration time.

Missed dose icon 528 is a circular icon displayed, for example, in red, and has a thick border. The missed dose icon 528 indicates that a patient failed to take a scheduled dose.

The colors and shapes of the icons 521 disclosed herein are not required and are not important. What is important is that a wrong time dose, on time dose, and missed dose each are represented by a unique icon or symbol. In addition, another row of icons can be displayed in each grid element to indicate the number of doses due, each icon representing a scheduled dose for a day.

In one embodiment, each of the grid elements in grid 520 are graphical user controls. A user may cause the computer to display more information about a particular day reflected in grid 502 by manipulating the day's respective grid element. For example, a user, using mouse 103 as an input device, moves a mouse cursor of calendar 500 onto the day's respective grid element and then clicks the mouse. In response, computer 101 displays a graphical time line with icons positioned to reflect when the drug was delivered.

FIG. 5C depicts an exemplary graphical time line. Time line 550 is a graphical image having a horizontal length that reflects one 24-hour day. One or more icons 562 each represent a dose delivered for a particular day. Each of the icons 562 are displayed along the horizontal axis 564 of time line 550 so that their respective positions along the horizontal axis of time line 550 reflects when they were delivered. One or more hour labels 566 indicate the time at which a dose was delivered. For example, icon 562 represents a dose that was delivered at approximately 8:00 a.m., as indicated by hour label 568.

In one embodiment, icons 562 may include icons for missed doses. Such icons may be displayed using a different pattern than those used to represent doses delivered on time. In addition, icons representing doses delivered at the wrong time can be displayed using a third pattern.

## COMPLIANCE INDEXES

Compliance information can also be provided in the form of compliance indexes. A compliance index is a set of one or more values that reflects the degree to which the actual delivery of a drug complies with the prescribed administration. A variety of compliance indexes may used.

For example, the compliance indexes may include a dosage-on-time index. The dosage-on-time index reflects the percent of doses that were delivered to the patient on time in a given period. For example, assume that a drug is prescribed to be administered three times a day, at 7:00 a.m., 3:00 p.m., and 11:00 p.m., plus or minus an hour. If for a given day the drug is in fact delivered twice at 8:00 a.m. and 6:00 p.m., then the dosage-on-time index for the day is thirty-three percent (33%).

A dose-per-day index reflects the percentage of prescribed doses that were at least delivered in a given period. In the previous example, the dose-per-index would be sixty-six percent (66%) because two out of three doses were delivered in the day.

A unit-per-day-index reflects what portion of the amount of a drug prescribed for a day was delivered to the patient. For example, 2000 mg may be prescribed, but 2200 mg may be delivered to the patient. Thus, the unit-per-day-index would be 110%.

The user may specify the period covered by a compliance index in a variety of ways. For example, a graphical user control list box may provide selectable list box items which each represent a period for which to generate a compliance index. One list box item specifies the last week, another the last two weeks, and another the previous month. In addition, the graphical user control text boxes can be configured to accept the beginning and end dates of a period.

Also, various techniques may be used to display compliance indexes to the user. Each index can be displayed as a numeral, or a graphic, such as a horizontal bar. The length of the bar would represent 100 percent, and a position of an indicator along the length would indicate a percent.

One or more compliance indexes may be presented in the form of a weekly dosing graph, as shown in FIG. 5C, or in other graph forms, such as a line, area, and histogram graph. In addition, a GUI may present a graphical user control through which a user may select the form of the graph for displaying compliance indexes. For example, a GUI may display a graphical user control list box containing list box items for each graph form. By selecting one of the list box items, a user specifies a graph form for displaying a compliance index.

Fig. 6 shows a score histogram graph according to an embodiment of the present invention. Score histogram graph 600 displays patient dosing scores in the form of a graph of "Time Span" versus "Score." The time span is selectable for a time range specified by the user. The score value represents a compliance index over, for example, the last 7, 14, 21, or 28 days, or a time span specified by the user.

Score histogram graph 600 contains one or more graphical bars, such as graphical bar 610. Each graphical bar is used to reflect a dosage score for a time period within the time span,

such as a day. To measure the graphical bars, score histogram graph 600 includes graphical score scale 604. The height of the graphical bars together with graphical score scale 604 indicate a dosage score for a particular time period. Graphical bar 610 reflects a score of 66%.

#### OTHER REPORTS

5 Other reports can be generated based on the foregoing information.

In particular, a Patient Dosing Report is generated based on data retrieved from the dispenser device 110. The data is displayed in a scrollable tabular grid. Displayed values include all medication events, dates, times, and dose sizes that are retrieved from the dispenser. Other non-medication events that are reported from the dispenser device to the  
10 computer 101 can be displayed at the option of the user. For example, when a user replaces a bottle in the dispenser, the dispenser device 110 reports a "bottle replaced" event to the computer 101. Such events can appear in the Patient Dosing Report.

As another example, a Patient Summary Report is generated. The report includes a header containing complete patient information such as Name, ID, Monitoring Dates, Drug,  
15 Brand, etc.

A Patient Summary Report, based on the merged data created in block 344 of FIG. 3, can be generated. The report summarizes data downloaded from multiple devices for the same patient.

A Summary of All Patients report presents a summary of all patients in grid form. The  
20 grid includes Name, ID, and Score for each patient. The grid may be sorted by any column. The Score value may be selected based on Doses Per Day or Time Of Day.

Preferably, the system provides a Print Preview function whereby the user can view any pages on the screen before they are printed.

#### PROGRAM STRUCTURE

25 Embodiments of the methods described further below may be implemented, for example, in one or more computer programs developed using Microsoft Visual Basic®. Preferably, the programs provide a multi-document interface whereby a user may view multiple documents simultaneously within the program. For example, the calendar dialog and medication event data dialogs described herein may be viewed at the same time.

30 In one embodiment, the program functions and method steps described above are organized in an application program using one or more pull-down menus, each of which has one or more menu options. Table 1 presents a hierarchy of menu options in one embodiment of such a program.

TABLE 1 -- MENU OPTIONS

|    |                 |
|----|-----------------|
| 35 | FILE            |
|    | New             |
|    | Open            |
|    | Save ...        |
|    | Save As ...     |
| 40 | Print Setup ... |

Print Preview ...

Print ...

Exit ...

DEVICE

5 Retrieve Dispenser Data

Program Dispenser

VIEW

Dosing Data

Dosing Calendar

10 Reports & Graphs ...

HELP

About

The application program may also provide confirmation dialogs that prompt the user to verify various functions, such as dosing, as they are performed and where appropriate.

15 In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

APPENDIX

# CycloTech Medication Monitoring Program

SangStat Medical Corporation

Produced by Glen Hamilton, Cyber Innovations Corporation

Code Listing From  
3/19/98

## Table Of Contents

---

|                          |     |
|--------------------------|-----|
| General.bas              | 1   |
| Comm.bas                 | 26  |
| Printing.bas             | 46  |
| Fax.bas                  | 57  |
| Calendar.bas             | 61  |
| frmMain.frm              | 76  |
| frmSplash.frm            | 85  |
| frmLogin.frm             | 86  |
| frmOptions.frm           | 88  |
| frmAbout.frm             | 92  |
| frmBrowser.frm           | 95  |
| frmTip.frm               | 98  |
| frmAllPatients.frm       | 100 |
| frmRecentDosingGraph.frm | 106 |
| frmDosingCalendar.frm    | 111 |
| frmPatientDosingRpt.frm  | 117 |
| frmReadDeviceData.frm    | 122 |
| frmPrint.frm             | 124 |
| frmDeviceDiagnostics.frm | 128 |
| frmFaxStatus.frm         | 133 |
| frmFaxSend.frm           | 134 |
| frmFaxLog.frm            | 141 |
| frmFaxEditGroups.frm     | 142 |
| frmFaxEditLocations.frm  | 144 |
| frmDeviceInitialize.frm  | 146 |
| frmGetDateTime.frm       | 151 |

---

Attribute VB\_Name = "modGeneral"  
Option Explicit

*'Declare DLL calls*

Declare Function OSWinHelp% Lib "user32" Alias "WinHelpA" (ByVal hWnd&, ByVal HelpFile\$, ByVal wCommand%, dwData As Any)  
Declare Function GetPrivateProfileInt Lib "kernel32" Alias "GetPrivateProfileIntA" (ByVal lpApplicationName As String, ByVal lpKeyName As String, ByVal nDefault As Long, ByVal lpFileName As String) As Long  
Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long  
Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long

*'Set up some temporary buffers for getting strings from DLL calls*

Public Const giBufSize1024 = 1024 *'set size of input buffer for strings*  
Public gsTempBuf As String *'input buffer for strings (defined at program start)*

Public giLatestOptionsTabSelected As Integer *'keep track of the tab that was last selected by user (goes back to it next time it is opened)*

Public gsLastStartDateChosen As String *'a starting plot date last selected by user*  
Public gsLastEndDateChosen As String *'an ending plot date last selected by user*  
Public gsLastDateSet As String *'a temp string used to pass dates back and forth to the calendar*  
Public gsDateDisplayFormat As String *'holds the user's choice for the displayed date format for dialogs and reports*  
Public gsTimeDisplayFormat As String *'holds the user's choice for the displayed time format for dialogs and reports*  
Public gsCustomLblPatientLastName As String *'replacement labels for the dialogs if exist in config file.*  
Public gsCustomLblPatientFirstName As String *'replacement labels for the dialogs if exist in config file.*  
Public gsCustomLblPatientID As String  
Public gsCustomLblTxCenter As String  
Public gsCustomLblDrug As String  
Public gsCustomLblOrgan As String  
Public gsLabelGridColumnCustom1 As String  
Public gsLabelGridColumnCustom2 As String  
Public gsLabelGridColumnCustom3 As String

*'This value is stored in device & indicates the version of data structure within the device.  
'This does not relate directly to the version of the host software because the host software version can change with meaning that the structure of the data in the device has changed.  
'This value should be increased when any kind of change occurs to the custom areas of the device such as changing the length of strings to accommodate new features. The purpose of this value is to let us read it back from a device and determine if newer host software is being used on a device programmed with another version.*

Public Const gsREV\_DATA\_STRUCTURE = "01"

*'There are 4 fields in the device containing 16 characters each. In the original device design, this was intended to contain 4 separate pieces of information.*

*'The length of each data type is as follows:*

Public Const giLEN\_REV\_DATA\_STRUCTURE = 2  
Public Const giLEN\_PATIENT\_NAME = 26  
Public Const giLEN\_ID = 11  
Public Const giLEN\_DRUG = 2  
Public Const giLEN\_TX\_CENTER = 18  
Public Const giLEN\_ORGAN = 2

Public Const giMaxDoseTimes = 4 *'the max number of prescribed dosing time (entry boxes)*

Public Const giDosesPerDayDefault = 2

Public gbPatientDataNotSaved As Boolean *'true once the data in memory has been saved (from device)*

Public gdTempDateTime As Double

Public giTempCya As Integer *'right, put this var in the form that uses it, can also be done*

Public giTempCreatinine As Single

Public gsTempCustomInfo As String

Public gsActiveFormName As String



```
Public giCurrentTip As Integer      'most recent tip number that was shown
Public gsWebStartingAddress As String 'url address and any associated password for web site
```

**Public Function ComputeIniSectionChecksum(ByVal sFileSpec As String, ByVal sSection As String)**

*'Read each line in the section name of an INI file that was passed here.*

*'Compute a unique value and pass back to caller*

*On Error GoTo 0 'rgn temp*

```
Dim IChecksumTally As Long, r As Integer, l As Long, i As Integer, iKey As Integer
Dim sLine As String
```

*'Get the names of all of the keys in this section.*

*'A null key field in above line loads all keys in that section*

```
Dim iStrSize As Integer, sTempBuf As String, iBufSize As Integer
```

```
Dim sKeyList(2000) As String      'make room for this many key names in this section
```

```
sTempBuf = Space$(16384)
```

```
iBufSize = 16384
```

```
l = GetPrivateProfileString(sSection, ByVal 0&, "", sTempBuf, iBufSize, sFileSpec)
```

```
r = ParseDelimString(Left$(sTempBuf, l), Chr$(0), sKeyList())      'put the key names in a list
```

```
For iKey = 1 To r
```

```
    sLine = GetINISetting(sFileSpec, sSection, sKeyList(iKey), "")
```

```
    For i = 1 To Len(sLine)
```

```
        IChecksumTally = IChecksumTally + (Asc(Mid$(sLine, i, 1)) * iKey)
```

```
    Next i
```

```
Next iKey
```

```
ICheckSumTally = IChecksumTally Mod 536870912      'a 29 bit number
```

```
ComputeIniSectionChecksum = IChecksumTally      'pass result back to caller
```

End Function

**Public Sub EventDelete(DataStruct As DeviceDataStruct, ByVal iIndex As Integer)**

*'Remove an event from the data structure. The index to the position is passed here.*

```
Dim i As Integer
```

*'it is not a valid index*

```
If iIndex < 1 Or iIndex > DataStruct.iEventData(0) Then Exit Sub
```

```
For i = iIndex To DataStruct.iEventData(0)      'move all events up one
```

```
    DataStruct.byteEventType(i) = DataStruct.byteEventType(i + 1)
```

```
    DataStruct.dEventData(i) = DataStruct.dEventData(i + 1)
```

```
    DataStruct.iEventData(i) = DataStruct.iEventData(i + 1)
```

```
Next i
```

```
DataStruct.iEventData(0) = DataStruct.iEventData(0) - 1      'decrement event count
```

```
gbPatientDataNotSaved = True      'set flag to indicate that the file has changed but not yet been saved
```

End Sub

## General.bas - EventInsert

3

**Public Sub EventInsert(DataStruct As DeviceDataStruct, ByVal iIndex As Integer, ByVal dDate As Double)**

*'Insert a new event into the data structure at the index location passed here. If the index = 0  
'then it probably indicates that a previous function could not find where to insert the date  
'in the structure. In this case, the event must be inserted at the beginning or the end of the  
'structure depending on the date.*

Dim i As Integer

```
If iIndex = 0 Then 'date was not found
  If dDate <= DataStruct.dEventData(1) Then
    iIndex = 1
  Else
    iIndex = DataStruct.iEventData(0) 'insert at last point
  End If
End If
```

```
If iIndex Then 'there are events in the structure
  For i = DataStruct.iEventData(0) To iIndex Step -1 'move all events down to make room for new one
    DataStruct.byteEventType(i + 1) = DataStruct.byteEventType(i)
    DataStruct.dEventData(i + 1) = DataStruct.dEventData(i)
    DataStruct.iEventData(i + 1) = DataStruct.iEventData(i)
    DataStruct.sUserData1(i + 1) = DataStruct.sUserData1(i)
    DataStruct.sUserData2(i + 1) = DataStruct.sUserData2(i)
    DataStruct.sUserData3(i + 1) = DataStruct.sUserData3(i)
  Next i
  Else
    iIndex = iIndex + 1
  End If
```

*'Now insert the new event*  
 DataStruct.iEventData(0) = DataStruct.iEventData(0) + 1 *'increment event count*  
 DataStruct.byteEventType(iIndex) = giEVENT\_USER\_DEFINED  
 DataStruct.dEventData(iIndex) = dDate  
 DataStruct.sUserData1(iIndex) = giTempCya *'put change in structure*  
 DataStruct.sUserData2(iIndex) = giTempCreatinine *'put change in structure*  
 DataStruct.sUserData3(iIndex) = gsTempCustomInfo *'put change in structure*

```
If iIndex = 1 Then 'there were no previous events until this one
  DataStruct.iEventData(iIndex) = 0
Else
  DataStruct.iEventData(iIndex) = DataStruct.iEventData(iIndex - 1)
End If
gbPatientDataNotSaved = True 'set flag to indicate that the file has changed but not yet been saved
```

End Sub

**Public Function FindPrescribedDoseSizeForSpecificDay(DataStruct As DeviceDataStruct, ByVal IDate As Long)**

*'Find the prescribed dose for the day that is passed here.  
'This is accomplished by looking for the most recent dose change  
'event that occurred on or prior to this date.*

```
Dim i As Integer, iIndex As Integer
iIndex = FindClosestDateInArray(DataStruct, IDate)
If iIndex = 0 Then 'all events are occurring after the date requested
  For i = 1 To DataStruct.iEventData(0) 'look through whole array if necessary
    If DataStruct.byteEventType(i) = giEVENT_DOSE_CHANGED Then
      FindPrescribedDoseSizeForSpecificDay = i 'DataStruct.iEventData(i)
    Exit For
  End If
Next i
```

```
Else 'an event date was found
  For i = iIndex To 1 Step -1
    If DataStruct.byteEventType(i) = giEVENT_DOSE_CHANGED Then
```

General.bas - FindPrescribedDoseSizeForSpec Day

4

```

        FindPrescribedDoseSizeForSpecificDay = i      'DataStruct.iEventData(i)
    Exit For
End If

Next i
End If

End Function

```

**Public Function CalcDayDoseScore\_OnTime(DataStruct As DeviceDataStruct, ByVal IStartingDate As Long) As**

*'Compute the dosing score for the day passed here  
 'This score tests to see if the doses taken was within the prescribed time range.  
 'Pass the score back to the caller as nearest whole percent.  
 'Index is the index in the array where computation is to start.  
 'It should already be set to the first event that occurred on that day.*

Dim I As Long, i As Integer, iTotalDoses As Integer  
 Dim iIndex As Integer, r As Integer

```

iIndex = FindClosestDateInArray(DataStruct, IStartingDate)      'returns 0 if date is not found
If iIndex Then 'an event was found
    Do 'look at all past events for the past iScoreDays
        If Int(DataStruct.dEventData(iIndex)) = IStartingDate Then
            'date still in range. ok to continue
            If DataStruct.byteEventType(iIndex) = giEVENT_DOSE_TAKEN Then 'this is a medication
                'Now test to see if time is within the daily prescribed range
                r = IsDoseWithinPrescribedTimeRange(DataStruct, iIndex) 'pass index to event time
                If r Then iTotalDoses = iTotalDoses + 1
            End If
            iIndex = iIndex + 1
        Else
            Exit Do
        End If
    Loop

    CalcDayDoseScore_OnTime = 100 * iTotalDoses / DataStruct.iDosesPerDay
End If

```

End Function

**Public Function CalcDayDoseScore\_AllDoses(DataStruct As DeviceDataStruct, ByVal IStartingDate As Long) As**

*'Compute the dosing score for the day passed here.  
 'Calculate for all doses taken on that day regardless of if they were taken on time or not.  
 'Pass the score back to the caller as nearest whole percent.  
 'Index is the index in the array where computation is to start.  
 'It should already be set to the first event that occurred on that day.*

Dim iTotalDoses As Integer, iIndex As Integer

```

iIndex = FindClosestDateInArray(DataStruct, IStartingDate)
If iIndex Then 'an event was found on this date
    Do 'look at all dosing events for this day
        If Int(DataStruct.dEventData(iIndex)) = IStartingDate Then
            'date still in range. ok to continue
            'this is a medication
            If DataStruct.byteEventType(iIndex) = giEVENT_DOSE_TAKEN Then iTotalDoses = iTotalDoses + 1
            iIndex = iIndex + 1
        Else
            Exit Do
        End If
    Loop

    CalcDayDoseScore_AllDoses = 100 * iTotalDoses / (DataStruct.iDosesPerDay)
End If

```

General.bas - CalcDayDoseScore\_AtIDose

5

End Function

**Public Function CalcDosesSumTakenOnSpecificDay(DataStruct As DeviceDataStruct, ByVal IStartingDate As I**

*'Computer the dosing total number of doses taken on a specific date  
'Note, this calculation does not take into consideration whether or not the dose  
'was taken within the prescribed time. This is all doses for a particular day  
'Pass the count back to the caller.*

Dim iTodayDoseCount As Integer, iIndex As Integer

iIndex = FindFirstMatchingDateInArray(DataStruct, IStartingDate)

If iIndex Then *'an event was found on this date*

Do *'look at all dosing events for this day*

If Int(DataStruct.dEventDate(iIndex)) = IStartingDate Then

*'date still in range, ok to continue*

*'this is a medication*

If DataStruct.byteEventType(iIndex) = giEVENT\_DOSE\_TAKEN Then iTodayDoseCount = iTodayDoseCount + 1

iIndex = iIndex + 1 *'goto next higher event in array*

*'exit if at end of array (prevents error)*

If UBound(DataStruct.dEventDate()) = iIndex Then Exit Do

*'exit if no data in array*

If iIndex > Int(DataStruct.iEventData(0)) Then Exit Do

Else

Exit Do

End If

Loop

CalcDosesSumTakenOnSpecificDay = iTodayDoseCount

End If

End Function

**Public Sub EraseDataInMemory(DataStruct As DeviceDataStruct)**

Dim i As Integer

*'clear out any data that may be in memory and initialize the arrays*

DataStruct.sPatientLastName = ""

DataStruct.sPatientFirstName = ""

DataStruct.sPatientID = ""

DataStruct.sDrug = ""

DataStruct.sOrgan = ""

DataStruct.sTxCenter = ""

DataStruct.sSerialNumber = ""

DataStruct.sFirmwareVer = ""

DataStruct.sDoseSize = ""

DataStruct.sPatientDataFileName = ""

For i = 0 To giMaxDoseTimes

DataStruct.dPrescribedDoseTime(i) = -1

Next i

DataStruct.iDosesPerDay = 0

DataStruct.sDoseResolution = ""

DataStruct.sMedRemaining = ""

Erase DataStruct.sScoreData

Erase DataStruct.iEventData

Erase DataStruct.dEventDate

Erase DataStruct.byteEventType *'erases all elements of a fixed array*

Erase DataStruct.sUserData1

Erase DataStruct.sUserData2

## General.bas - EraseDataInMemory

6

Erase DataStruct.sUserData3

DataStruct.lDeviceInitDate = 0  
 DataStruct.sBatteryChangeTimer = ""  
 DataStruct.sDoseLockoutHours = ""

DataStruct.bErrorFatal = False  
 DataStruct.bErrorNonFatal = False  
 DataStruct.bErrorDoseSize = False  
 DataStruct.bErrorMedRemaining = False  
 DataStruct.bErrorMemoryFull = False  
 DataStruct.bErrorsExist = False  
 DataStruct.bErrorBrownOut = False  
 DataStruct.dLastDownloadDate = 0

gbPatientDataNotSaved = False

End Sub

## Public Sub CreateTxtSummaryFile()

*'This routine creates a temp text file in the "fax" subdirectory  
 'This will allow the information to be faxed as a text document.*

Dim i As Integer, r As Integer, sFileSpec As String, IErrorCode As Long  
 Dim sLblName As String, sLblID As String, sLblTxCenter As String, sLblDrug As String, sLblOrgan As String

*'Get rid of the previous temporary file.*

sFileSpec = App.Path + "\Vaxes\temp.txt"

sFileSpec = App.Path + "\Vaxes\" + PAT\_DATA.sPatientLastName + ", " + PAT\_DATA.sPatientFirstName + " " + PAT\_DATA.sPatientID + ".txt"

r = FileExists(sFileSpec, IErrorCode)

If r Then Kill sFileSpec

sLblName = gsCustomLblPatientLastName

sLblID = gsCustomLblPatientID

sLblTxCenter = gsCustomLblTxCenter

sLblDrug = gsCustomLblDrug

sLblOrgan = gsCustomLblOrgan

Open sFileSpec For Output Shared As #1

Print #1, sLblName + ": " + PAT\_DATA.sPatientLastName + " " + PAT\_DATA.sPatientFirstName

Print #1, sLblID + ": " + PAT\_DATA.sPatientID

Print #1, sLblTxCenter + ": " + PAT\_DATA.sTxCenter

Print #1, sLblDrug + ": " + PAT\_DATA.sDrug

Print #1, sLblOrgan + ": " + PAT\_DATA.sOrgan

Print #1,

Print #1, "Device Serial Number: " + PAT\_DATA.sSerialNumber

Print #1, "FirmWare Version: " + PAT\_DATA.sFirmwareVer

Print #1, "Last Download Date: " + Format\$(PAT\_DATA.dLastDownloadDate, gsDateDisplayFormat)

Close #1

*'rgh ensure that the complete file is printed*

End Sub

## General.bas - FileExists

7

**Function FileExists(ByVal sPath As String, IErrorCode As Long) As Integer**

*\* Check for existence of a file by attempting an OPEN.  
 \* Return true (-1) if exists else return False (0) or error condition  
 \* Note that since this function tries to open a file, an error could  
 \* return to caller if file is there but in use by another application.*

```
Dim X As Integer
```

```
X = FreeFile
```

```
On Error Resume Next
```

```
Open sPath For Input As X
```

```
Close X
```

```
If Err = 0 Then
```

```
FileExists = True
```

```
IErrorCode = 0      'clear error code
```

```
Else
```

```
FileExists = False  'set flag for error
```

```
IErrorCode = Err    'pass error back to caller
```

```
End If
```

```
End Function
```

**Public Function GetINISetting(sFileSpec As String, sSection As String, sKeyField As String, sDefault As String**

```
Dim IStrSize As Integer, sTempBuf As String, iBufSize As Integer
```

```
sTempBuf = Space$(1024)
```

```
iBufSize = 1024
```

```
IStrSize = GetPrivateProfileString(sSection, sKeyField, sDefault, sTempBuf, iBufSize, sFileSpec)
```

```
If IStrSize Then
```

```
GetINISetting = Trim$(Left$(sTempBuf, IStrSize))
```

```
Else
```

```
GetINISetting = sDefault
```

```
End If
```

```
End Function
```

**Public Function GetPatientDataFromDisk(ByVal sFileSpec As String, DataStruct As DeviceDataStruct, IErrorRe**

*\*Get all of the patient data from the file on disk and place into memory.*

*\*The filename that is passed here must be a valid patient file and verified  
 \*by the calling procedure.*

```
Dim sSection As String, i As Integer, sTemp As String, r As Integer
```

```
Dim IFileChecksum As Long, IChecksumTally As Long
```

```
On Error GoTo GetPatientDataFromDisk_Error
```

*\*Read the file and calculate the checksum.*

```
IFileChecksum = ComputeIniSectionChecksum(sFileSpec, "Device Data")
```

```
ICheckSumTally = GetINISetting(sFileSpec, "General", "Device Data Validation", 0)
```

```
If IFileChecksum <> IChecksumTally Then
```

```
IErrorReturn = ERR_DATA_CHECKSUM
```

```
Exit Function
```

```
End If
```

```
IFileChecksum = ComputeIniSectionChecksum(sFileSpec, "Event Data")
```

```
ICheckSumTally = GetINISetting(sFileSpec, "General", "Event Data Validation", 0)
```

```
If IFileChecksum <> IChecksumTally Then
```

```
IErrorReturn = ERR_DATA_CHECKSUM
```

```
Exit Function
```

```
End If
```

## General.bas - GetPatientDataFromDisk

8

```

IFileChecksum = ComputeIniSectionChecksum(sFileSpec, "Device Error Flags")
IChecksumTally = GetINISetting(sFileSpec, "General", "Device Error Flags Validation", 0)
If IFileChecksum <> IChecksumTally Then
    IErrorReturn = ERR_DATA_CHECKSUM
    Exit Function
End If

sSection = "Device Error Flags"
DataStruct.bErrorFatal = CBool(GetINISetting(sFileSpec, sSection, "Fatal", False))
DataStruct.bErrorNonFatal = CBool(GetINISetting(sFileSpec, sSection, "Non Fatal", False))
DataStruct.bErrorDoseSize = CBool(GetINISetting(sFileSpec, sSection, "Dose Size", False))
DataStruct.bErrorMedRemaining = CBool(GetINISetting(sFileSpec, sSection, "Med Remaining", False))
DataStruct.bErrorMemoryFull = CBool(GetINISetting(sFileSpec, sSection, "Memory Full", False))
DataStruct.bErrorBrownOut = CBool(GetINISetting(sFileSpec, sSection, "Brownout", False))

sSection = "Device Data"
EraseDataInMemory DataStruct

sTemp = GetINISetting(sFileSpec, sSection, "Device Init Date", 0)
If IsDate(sTemp) Then
    DataStruct.lDeviceInitDate = DateValue(sTemp)
End If

sTemp = GetINISetting(sFileSpec, sSection, "Events Ref Date Time", 0)
If IsDate(sTemp) Then
    DataStruct.dDeviceRefDateTime = DateValue(sTemp)
End If

sTemp = GetINISetting(sFileSpec, sSection, "Last Download Date", 0)
If IsDate(sTemp) Then
    DataStruct.dLastDownloadDate = DateValue(sTemp)
End If

DataStruct.sPatientLastName = GetINISetting(sFileSpec, sSection, "Last Name", "")
DataStruct.sPatientFirstName = GetINISetting(sFileSpec, sSection, "First Name", "")
DataStruct.sPatientID = GetINISetting(sFileSpec, sSection, "Patient ID", "")
DataStruct.sTxCenter = GetINISetting(sFileSpec, sSection, "Tx Center", "")
i = CInt(GetINISetting(sFileSpec, sSection, "Organ Reference Number", 0))
If i And i <= UBound(gsOrganNames) Then DataStruct.sOrgan = gsOrganNames(i)
i = CInt(GetINISetting(sFileSpec, sSection, "Drug Reference Number", 0))
If i And i <= UBound(gsDrugNames) Then DataStruct.sDrug = gsDrugNames(i)
DataStruct.sSerialNumber = GetINISetting(sFileSpec, sSection, "Serial Number", "")
DataStruct.sFirmwareVer = GetINISetting(sFileSpec, sSection, "Firmware Version", "")
DataStruct.sDoseSize = GetINISetting(sFileSpec, sSection, "Dose Size", "")
DataStruct.lDosesPerDay = CInt(GetINISetting(sFileSpec, sSection, "Doses Per Day", 0))
DataStruct.sDoseResolution = GetINISetting(sFileSpec, sSection, "Dose Resolution", "")
DataStruct.sMedRemaining = GetINISetting(sFileSpec, sSection, "Medication Remaining", "")
DataStruct.sBatteryChangeTimer = GetINISetting(sFileSpec, sSection, "Battery Change Timer", "")
DataStruct.sDoseLockoutHours = GetINISetting(sFileSpec, sSection, "Lockout Hours Between Doses", "")

For i = 1 To 14
    DataStruct.sScoreData(i) = GetINISetting(sFileSpec, sSection, "Patient Score Data " + CStr(i), "")
Next i

For i = 1 To giMaxDoseTimes
    sTemp = GetINISetting(sFileSpec, sSection, "Prescribed Dose Time " + CStr(i), "-1")
    DataStruct.dPrescribedDoseTime(i) = -1 ' default value
    If IsDate(sTemp) Then DataStruct.dPrescribedDoseTime(i) = CDate(sTemp)
Next i

DataStruct.iEventData(0) = CInt(GetINISetting(sFileSpec, "Event Data", "Event Count", 0))

Dim sTempList(10) As String
For i = 1 To DataStruct.iEventData(0)

```

## General.bas - GetPatientDataFromDisk

9

```

sTemp = GetINISetting(sFileSpec, "Event Data", CStr(i), "")
r = ParseDelimString(sTemp, ",", sTempList())
DataStruct.dEventData(i) = CDate(sTempList(1))
Select Case Trim$(LCase$(sTempList(2)))
    Case "dose taken"
        DataStruct.byteEventType(i) = giEVENT_DOSE_TAKEN
        DataStruct.iEventData(i) = sTempList(3)
    Case "dose change"
        DataStruct.byteEventType(i) = giEVENT_DOSE_CHANGED
        DataStruct.iEventData(i) = sTempList(3)
    Case "custom event"
        DataStruct.byteEventType(i) = giEVENT_USER_DEFINED
        DataStruct.iEventData(i) = sTempList(3)
End Select
DataStruct.sUserData1(i) = sTempList(4)
DataStruct.sUserData2(i) = sTempList(5)
DataStruct.sUserData3(i) = sTempList(6)
Next i

' bErrorFatal As Boolean      'true if this flag was set in the returned flags string
' bErrorNonFatal As Boolean   'true if this flag was set in the returned flags string
' bErrorDoseSize As Boolean    'true if this flag was set in the returned flags string
' bErrorMedRemaining As Boolean 'true if this flag was set in the returned flags string
' bErrorMemoryFull As Boolean  'true if this flag was set in the returned flags string
' bErrorBrownOut As Boolean    'true if this flag was set in the returned flags string
' bErrorsExist As Boolean      '(1 byte) Bits are set if various errors have occurred and have not
GetPatientDataFromDisk = True 'return success flag to caller

GetPatientDataFromDisk_Exit:
Exit Function

GetPatientDataFromDisk_Error:
!ErrorReturn = Err
Resume GetPatientDataFromDisk_Exit

End Function

```

**Public Sub GetProgramPreferences()**

```

'Load the program and user preferences into the global variables
Dim iStrSize As Integer, i As Integer, sFileSpec As String, r As Integer
Dim sSection As String
sSection = "Preferences"
gsDateDisplayFormat = GetINISetting(gsAppIniFileSpec, sSection, "Date Display Format", "Short Date")
gsTimeDisplayFormat = GetINISetting(gsAppIniFileSpec, sSection, "Time Display Format", "Short Time")
gsngComplianceTimeRange = CSng(GetINISetting(gsAppIniFileSpec, sSection, "Compliance Time Range", "2"))

sSection = "Custom Settings"
'Get any custom field labels that may be in the INI file. If none exist the set some defaults here.
gsCustomLblPatientLastName = GetINISetting(gsAppIniFileSpec, sSection, "Last Name Label", "")
If gsCustomLblPatientLastName = "" Then gsCustomLblPatientLastName = "Last Name"

gsCustomLblPatientFirstName = GetINISetting(gsAppIniFileSpec, sSection, "First Name Label", "")
If gsCustomLblPatientFirstName = "" Then gsCustomLblPatientFirstName = "First Name"

gsCustomLblPatientID = GetINISetting(gsAppIniFileSpec, sSection, "Patient ID Label", "")
If gsCustomLblPatientID = "" Then gsCustomLblPatientID = "Patient ID"

gsCustomLblTxCenter = GetINISetting(gsAppIniFileSpec, sSection, "TX Center Label", "")
If gsCustomLblTxCenter = "" Then gsCustomLblTxCenter = "TX Center"

gsCustomLblDrug = GetINISetting(gsAppIniFileSpec, sSection, "Drug Label", "")
If gsCustomLblDrug = "" Then gsCustomLblDrug = "Drug"

```



## General.bas - GetProgramPreferences

10

```

gsCustomLblOrgan = GetINISetting(gsAppIniFileSpec, sSection, "Organ Label", "")
If gsCustomLblOrgan = "" Then gsCustomLblOrgan = "Organ"

gsLabelGridColumnCustom1 = GetINISetting(gsAppIniFileSpec, sSection, "Grid Column 1", "")
If gsLabelGridColumnCustom1 = "" Then gsLabelGridColumnCustom1 = "CYA Level (ng/ml)"

gsLabelGridColumnCustom2 = GetINISetting(gsAppIniFileSpec, sSection, "Grid Column 2", "")
If gsLabelGridColumnCustom2 = "" Then gsLabelGridColumnCustom2 = "Creatinine (mg/dl)"

gsLabelGridColumnCustom3 = GetINISetting(gsAppIniFileSpec, sSection, "Grid Column 3", "")
If gsLabelGridColumnCustom3 = "" Then gsLabelGridColumnCustom3 = "Custom"

'Get the list of most recently used files to the menu
For i = 1 To frmMain.mnuFileMRU.UBound
    frmMain.mnuFileMRU(i).Tag = GetINISetting(gsAppIniFileSpec, "Recent Files", CStr(i), "")
    If frmMain.mnuFileMRU(i).Tag <> "" Then
        frmMain.mnuFileMRU(i).Visible = True
        'strip the filespec away from the tag and put into the caption for display purposes
        r = GetFileNameFromSpec(frmMain.mnuFileMRU(i).Tag, sFileSpec) 'hold the name of the file
        frmMain.mnuFileMRU(i).Caption = sFileSpec
        frmMain.mnuFileBar6.Visible = True
    End If
Next i

'Get last values for the Fax control that was last set by user
sSection = "User Selections"
With FAX_DATA
    .sSenderName = GetINISetting(gsFaxFileSpec, sSection, "Sender Name", "")
    .sSenderCompany = GetINISetting(gsFaxFileSpec, sSection, "Sender Company", "")
    .sSenderVoiceNumber = GetINISetting(gsFaxFileSpec, sSection, "Sender Voice Number", "")
    .sSenderFaxNumber = GetINISetting(gsFaxFileSpec, sSection, "Sender Fax Number", "")
    .sFaxID = GetINISetting(gsFaxFileSpec, sSection, "Fax ID", "")
    .sDialPrefix = GetINISetting(gsFaxFileSpec, sSection, "Dial Prefix", "")
    .iRetries = CInt(GetINISetting(gsFaxFileSpec, sSection, "Retries", "0"))
    .iRetryInterval = CInt(GetINISetting(gsFaxFileSpec, sSection, "Retry Interval", "1"))
    .bFaxResolution = GetINISetting(gsFaxFileSpec, sSection, "Resolution", "0")
End With

'Get the Drug types from file and place in global list
sSection = "Transplant Centers"
TxCenters(0) = GetINISetting(gsAppIniFileSpec, sSection, "Count", "0")
For i = 1 To TxCenters(0)
    TxCenters(i) = GetINISetting(gsAppIniFileSpec, sSection, CStr(i), "0")
Next i

'Get the Drug types from file and place in global list
sSection = "Drugs"
gsDrugNames(0) = GetINISetting(gsAppIniFileSpec, sSection, "Count", "0")
For i = 1 To gsDrugNames(0)
    gsDrugNames(i) = GetINISetting(gsAppIniFileSpec, sSection, CStr(i), "0")
Next i

'Get the Drug types from file and place in global list
sSection = "Organs"
gsOrganNames(0) = GetINISetting(gsAppIniFileSpec, sSection, "Count", "0")
For i = 1 To gsOrganNames(0)
    gsOrganNames(i) = GetINISetting(gsAppIniFileSpec, sSection, CStr(i), "0")
Next i

giCurrentTip = CInt(GetINISetting(gsAppIniFileSpec, "Options", "Current Tip", 1))

'Get settings of calendar form
CAL_DEFAULTS.chkDosesMissed = CByte(GetINISetting(gsAppIniFileSpec, "Calendar Settings", "chkDosesMissed", 1))
CAL_DEFAULTS.chkDosesNotComplied = CByte(GetINISetting(gsAppIniFileSpec, "Calendar Settings", "chkDosesNotComplied", 1))

```

## General.bas - GetProgramPreferences

11

```

CAL_DEFAULTS.chkDosesTaken = CByte(GetINISetting(gsAppIniFileSpec, "Calendar Settings", "chkDosesTaken", 0))
CAL_DEFAULTS.chkDoseChanged = CByte(GetINISetting(gsAppIniFileSpec, "Calendar Settings", "chkDoseChanged", 1))

'Get Settings of Patient summary form
PAT_SUM_DEFAULTS.cmboDataToView = CByte(GetINISetting(gsAppIniFileSpec, "Patient Summary Settings", "cmboDataToView", 1))
PAT_SUM_DEFAULTS.cmboChartType = CByte(GetINISetting(gsAppIniFileSpec, "Patient Summary Settings", "cmboChartType", 1))

'Get the web address for the browser
gsWebStartingAddress = "http://"
gsWebStartingAddress = gsWebStartingAddress + GetINISetting(gsAppIniFileSpec, "Web Data", "User Name", "") + ":"
gsWebStartingAddress = gsWebStartingAddress + GetINISetting(gsAppIniFileSpec, "Web Data", "Access", "") + "@"
gsWebStartingAddress = gsWebStartingAddress + GetINISetting(gsAppIniFileSpec, "Web Data", "URL", "")

End Sub

```

## Sub Main()

```

Dim l As Long, r As Integer, i As Integer, sMSG As String, sTemp As String, dTime As Double
Dim bBrowserFound As Boolean, lLastAccessDate As Long, lNextReminderDate As Long

'Initialize some application settings
gsAppIniFileSpec = App.Path + "\\" + "CycloTech.ini"
gsFaxFileSpec = App.Path + "\Fax.ini"
gsTempBuf = Space$(1024)
gbCommOK = 99 'set to some value other than true or false to properly initialize the dialog

frmSplash.Show
frmSplash.Refresh
dTime = Now 'get the time value of now

Wait 0.75
frmLogin.Show vbModal
If Not frmLogin.OK Then End 'Login Failed so exit app
Unload frmLogin
DoEvents

'rgn note that the debug flag is turned on and the fax icon is hidden
l = Shell(App.Path + "Faxman32.exe /D /H", 1) 'start the fax server
i = Shell("Faxman32.exe /H", 1) 'start the fax server
Load frmMain
Set gcFax = frmMain.FaxMan1

'If browser feature is turned on in the ini file, then activate item on the menu.
'See if we should allow access to visit Sangstat on the Internet
r = CBool(GetINISetting(gsAppIniFileSpec, "Web Data", "Active", "False"))
If r = False Then frmMain.mnuAccessWebSite.Visible = False 'no key found in ini file

Comm_InitializeCommPort 'initialize the comm port from INI file settings
GetProgramPreferences
EraseDataInMemory PAT_DATA
EraseDataInMemory TEMP_DATA

'Set some menu items
frmMain.mnuFileSave.Enabled = False
frmSplash.ZOrder

Do
    If CDBl(Now) > dTime + 0.00005 Then Exit Do 'wait for a minimum amount of time before unloading splash screen
    DoEvents
Loop

```

## General.bas - Main

12

Unload frmSplash

frmMain.Show  
SetPrinterIcon False, ""

*'See if we should shown tips at startup*  
r = CBool(GetINISetting(gsAppIniFileSpec, "Options", "Show Tips at Startup", True))  
If r Then frmTip.Show

*'See if we should remind user to visit Sangstat on the Internet*  
If frmMain.mnuAccessWebSite.Visible = True Then *'user must have menu selection on to access web*  
*'Try to find a browser by looking in different locations in the registry*  
frmMain.MhIni1.Key = ClassesRoot  
frmMain.MhIni1.EntrySection = "HTTP\shell\open\command"  
frmMain.MhIni1.EntryItem = "" *'gets the default value*  
frmMain.MhIni1.Action = 13 *'get registry key*  
If Len(Trim(frmMain.MhIni1.EntryValue)) > 0 Then  
    bBrowserFound = True *'Looks like a value is there*  
End If

frmMain.MhIni1.Key = LocalMachine  
frmMain.MhIni1.EntrySection = "SOFTWARE\Classes\HTTP\shell\open\command"  
frmMain.MhIni1.EntryItem = "" *'gets the default value*  
frmMain.MhIni1.Action = 13 *'get registry key*  
If Len(Trim(frmMain.MhIni1.EntryValue)) > 0 Then  
    bBrowserFound = True *'Looks like a value is there*  
End If

If bBrowserFound Then  
    iLastAccessDate = 0  
    sTemp = GetINISetting(gsAppIniFileSpec, "Web Data", "Last Web Visit Date", "")  
    If IsDate(sTemp) Then iLastAccessDate = DateValue(sTemp)  
  
    iNextReminderDate = 0  
    sTemp = GetINISetting(gsAppIniFileSpec, "Web Data", "Next Web Visit Reminder Date", "")  
    If IsDate(sTemp) Then iNextReminderDate = DateValue(sTemp)

*'On this line, if L is negative then it indicates that the user chose to not connect*  
*'the last time he/she was reminded. In this case we wait a much*  
*'shorter period of time before reminding them again.*  
If DateValue(Now) >= iNextReminderDate Then *'it's been too long since the user was last on the web.*

If iLastAccessDate > 0 Then  
    sMSG = "You last connected to our internet web site on " + sTemp + ". "  
Else  
    sMSG = "You have not yet connected to our internet web site. "  
End If

sMSG = sMSG + "There may be a program update or other valuable information there."  
sMSG = sMSG + vbCrLf + vbCrLf + "Would you like to connect to the web site now? (you must already have web access available)"

Beep  
r = MsgBox(sMSG, vbQuestion + vbYesNo + vbDefaultButton2, "Internet Connection Reminder")  
*'regardless of the answer to the next question. set a minimum time to ask user again.*  
*'If user actually connects to internet, then this time is overwritten with a longer one by the browser.*  
SaveINISetting gsAppIniFileSpec, "Web Data", "Next Web Visit Reminder Date", Format\$(Now + 15, "Medium Date")

If r = vbYes Then  
    Call LogonToWebSite

End If

End If

End If

End Sub

## General.bas - LogonToWebSite

13

**Public Sub LogonToWebSite()**

```

'Visit Sangstat on the Internet
'Dim frmB As New frmBrowser
'Load frmBrowser
frmBrowser.StartingAddress = gsWebStartingAddress
DoEvents 'allow time to paint
DoEvents 'allow time to paint
frmBrowser.Refresh
frmBrowser.Show
End Sub

```

**Public Function OpenPatientData(ByVal sFileSpec As String) As Integer**

*'Open patient data file and load to memory*

*'Return a true if load was successful, false if not, and vbCancel if user cancelled*

```

Dim r As Integer, sTemp As String, IErrorCode As Long
On Error GoTo OpenPatientData_Error

```

```

r = ValidatePatientDataSaved 'make sure any device data has first been saved
If r = vbCancel Then Exit Function

```

*'Get a filename from the common dialog*

*'Setup the common dialog control prior to showing it*

```

With frmMain.dlgCommonDialog

```

```

.Flags = cdlOFNOverwritePrompt Or cdlOFNPathMustExist Or cdlOFNExplorer Or cdlOFNExtensionDifferent Or

```

```

.cdlOFNNoReadOnlyReturn Or cdlOFNHideReadOnly Or cdlOFNFileMustExist

```

```

.CancelError = True 'generate error if CANCEL button is pressed

```

```

.InitDir = App.Path + "\Patient Data"

```

```

.Filter = "CycloTech Data File *.cpd|.cpd"

```

```

.DialogTitle = "Open Patient Data File"

```

```

.DefaultExt = ".CPD" 'append "Structure" extension when saving.

```

```

If sFileSpec <> "" Then .filename = sFileSpec

```

```

.ShowOpen 'Open dialog

```

```

End With

```

*'Now get the data from file*

```

frmMain.MousePointer = vbHourglass

```

```

DoEvents

```

```

r = GetPatientDataFromDisk(frmMain.dlgCommonDialog.filename, PAT_DATA, IErrorCode)

```

```

If r <> True Then

```

```

If IErrorCode = ERR_DATA_CHECKSUM Then

```

```

sTemp = "The contents of the data file have changed since it was last saved. "

```

```

sTemp = sTemp + "This could be due to a corrupt file, but is more likely that the file was manually changed."

```

```

sTemp = sTemp + vbCrLf + vbCrLf + "The file will not be loaded."

```

```

Beep

```

```

MsgBox sTemp, vbCritical, "File Contents Changed"

```

```

Else

```

```

MsgBox "An error occurred while retrieving data from the file. It was not read.", vbExclamation, "Error In File - " + Error(r)

```

```

End If

```

```

End If

```

```

PAT_DATA.sPatientDataFileName = frmMain.dlgCommonDialog.filename

```

```

OpenPatientData = True

```

```

r = GetFileNameFromSpec(PAT_DATA.sPatientDataFileName, sTemp) 'hold the name of the file

```

```

'UpdateRecentFileMenu sFileSpec

```

```

UpdateRecentFileMenu sTemp

```

```

frmMain.mnuFileSave.Enabled = True

```

```

OpenPatientData_Exit:

```

```

On Error GoTo 0

```

```

RefreshAllOpenForms

```

```

frmMain.MousePointer = vbDefault

```

```

Exit Function

```

## General.bas - OpenPatientData

14

```

OpenPatientData_Error:
If Err = cdCancel Then      'cancel button was pressed in dialog
    OpenPatientData = vbCancel
    Resume OpenPatientData_Exit
Else
    Beep
    MsgBox "The CycloTech Data file contains invalid data and can not be read.", vbExclamation, "Invalid Data File - " + Error
    PAT_DATA.sPatientDataFileName = ""
    frmMain.mnuFileSave.Enabled = False
End If
OpenPatientData = False
Resume OpenPatientData_Exit    'exit anyway for now

```

End Function

## Public Sub PopulateDeviceDiagDialog(DataStruct As DeviceDataStruct, SourceForm As Form)

*'There are two possible dialogs that have the same controls on them.**'This common procedure will populate both*

Dim i As Integer

With SourceForm

*'Show custom labels from config file if there were any*

.Label1(3) = gsCustomLblPatientLastName

.Label1(1) = gsCustomLblPatientFirstName

.Label1(5) = gsCustomLblPatientID

.Label1(6) = gsCustomLblTxCenter

.Label1(7) = gsCustomLblDrug

.Label1(0) = gsCustomLblOrgan

If TypeOf .txtDrug Is SSPanel Then

.txtDrug = DataStruct.sDrug

ElseIf TypeOf .txtDrug Is ComboBox Then *'it is a list box*

.txtDrug.Clear

For i = 1 To gsDrugNames(0) *'fill the drugs list box with available choices*

.txtDrug.AddItem gsDrugNames(i)

Next i

For i = 0 To .txtDrug.ListCount - 1

If .txtDrug.List(i) = DataStruct.sDrug Then

.txtDrug.ListIndex = i

Exit For

End If

Next i

End If

If TypeOf .txtOrgan Is SSPanel Then

.txtOrgan = DataStruct.sOrgan

ElseIf TypeOf .txtOrgan Is ComboBox Then *'it is a list box*

.txtOrgan.Clear

For i = 1 To gsOrganNames(0) *'fill the drugs list box with available choices*

.txtOrgan.AddItem gsOrganNames(i)

Next i

For i = 0 To .txtOrgan.ListCount - 1

If .txtOrgan.List(i) = DataStruct.sOrgan Then

.txtOrgan.ListIndex = i

Exit For

End If

Next i

End If

.txtPatientLastName = DataStruct.sPatientLastName

## General.bas - PopulateDeviceDiagDialog

15

```

.txtPatientFirstName = DataStruct.sPatientFirstName
.txtPatientID = DataStruct.sPatientID
.txtTxCenter = DataStruct.sTxCenter
.txtSerialNumber = DataStruct.sSerialNumber
.txtDoseSize = DataStruct.sDoseSize

.txtPatientLastName.SetFocus

If DataStruct.iEventData(0) Then
    .txtEventCount = " " + CStr(DataStruct.iEventData(0)) + " "
Else
    .txtEventCount = ""
End If

For i = 1 To giMaxDoseTimes
    If DataStruct.dPrescribedDoseTime(i) >= 0 Then .txtDoseTime(i) = Format$(DataStruct.dPrescribedDoseTime(i),
        gsTimeDisplayFormat)
Next i

.txtDosesPerDay = CStr(DataStruct.iDosesPerDay)
.txtDoseResolution = DataStruct.sDoseResolution
.txtDoseLockoutHours = DataStruct.sDoseLockoutHours

If DataStruct.iDeviceInitDate Then
    .txtDeviceStarted = " " + Format$(CDate(DataStruct.iDeviceInitDate), "Medium Date")
End If

.txtMedicationRemaining = " " + DataStruct.sMedRemaining
.txtBatteryChangeTimer = " " + DataStruct.sBatteryChangeTimer
.txtFirmwareVer = " " + DataStruct.sFirmwareVer

'set indicators for error flags
If DataStruct.bErrorFatal Then
    .imgFatal.Picture = .imgError.Picture
Else
    .imgFatal.Picture = .imgNoError.Picture
End If

If DataStruct.bErrorNonFatal Then
    .imgNonFatal.Picture = .imgError.Picture
Else
    .imgNonFatal.Picture = .imgNoError.Picture
End If

If DataStruct.bErrorDoseSize Then
    .imgDoseSize.Picture = .imgError.Picture
Else
    .imgDoseSize.Picture = .imgNoError.Picture
End If

If DataStruct.bErrorMedRemaining Then
    .imgMedRemaining.Picture = .imgError.Picture
Else
    .imgMedRemaining.Picture = .imgNoError.Picture
End If

If DataStruct.bErrorMemoryFull Then
    .imgMemoryFull = .imgError.Picture
Else
    .imgMemoryFull.Picture = .imgNoError.Picture
End If

If DataStruct.bErrorBrownOut Then
    .imgBrownOut = .imgError.Picture
Else
    .imgBrownOut.Picture = .imgNoError.Picture
End If

End With

```

## General.bas - PopulateDeviceDialog

16

End Sub

**Public Sub RefreshAllOpenForms()**

Dim r As Integer

*'if any of these forms are open at the time a new file is loaded,  
'then refresh them.*

For r = 0 To Forms.Count - 1

Select Case Forms(r).Name

Case "frmPatientDosingReport"

frmPatientDosingReport.UpdatefrmPatientDosingReportHeader

frmPatientDosingReport.UpdatePatientGridDisplay

Case "frmDosingCalendar"

If PAT\_DATA.dEventDate(PAT\_DATA.iEventData(0)) > 0 Then frmDosingCalendar.Calendar.Date = CVDDate(PAT\_DATA.  
dEventDate(PAT\_DATA.iEventData(0)))

UpdateCalendar

Case "frmPrint"

RefreshPreview

Case "frmPatientSummary"

frmPatientSummary.UpDatefrmPatientSummaryHeader

frmPatientSummary.cmboDateSelection\_Click

'frmPatientSummary.UpdatePatientDosingGraph

Case "frmDeviceInitialize"

PopulateDeviceCommDialog PAT\_DATA, frmDeviceInitialize

Case "frmReadDeviceData"

PopulateDeviceCommDialog PAT\_DATA, frmReadDeviceData

End Select

Next r

End Sub

**Public Sub SetPrinterIcon(bEnable As Boolean, sTip As String)**

On Error Resume Next

frmMain.mnuFilePrint.Enabled = bEnable

If sTip = "" Then

frmMain.mnuFilePrint.Caption = "Print..."

Else

frmMain.mnuFilePrint.Caption = sTip

End If

frmMain.tbToolBar.Buttons.Item(5).Enabled = bEnable

frmMain.tbToolBar.Buttons.Item(5).ToolTipText = sTip

*'If the active form is not the print form, then keep the name of the*

*'form in the key property of the icon. This is so that the print*

*'form will know what kind of information to display and print.*

If frmMain.ActiveForm.Name &lt;&gt; "frmPrint" Then gsActiveFormName = frmMain.ActiveForm.Name

On Error GoTo 0

End Sub

## General.bas - UpDateRecentFileMenu

17

**Public Sub UpDateRecentFileMenu(ByVal sFileSpec As String)**

*'Add the newest FileName to the menulist and move the other ones down.*  
*On Error GoTo 0*

Dim bDuplicateFound As Boolean, i As Integer, r As Integer, sFileName As String  
 r = GetFileNameFromSpec(sFileSpec, sFileName) *'hold the name of the file*

With frmMain

For i = 1 To .mnuFileMRU.UBound - 1

If LCase\$(.mnuFileMRU(i).Caption) = LCase\$(sFileName) Then *'remove any duplicates that might appear*  
     .mnuFileMRU(i).Caption = ""  
     .mnuFileMRU(i).Tag = ""  
     bDuplicateFound = True

End If

Next i

For i = .mnuFileMRU.UBound - 1 To 1 Step -1

If .mnuFileMRU(i).Caption <> "" Then *'contains a filename ok to shift down*  
     .mnuFileMRU(i + 1).Caption = .mnuFileMRU(i).Caption *'holds filename only for display purposes*  
     .mnuFileMRU(i + 1).Tag = .mnuFileMRU(i).Tag *'holds the filespec*  
     .mnuFileMRU(i + 1).Visible = True

Else

    .mnuFileMRU(i + 1).Visible = False

End If

Next i

.mnuFileMRU(1).Tag = sFileSpec  
 .mnuFileMRU(1).Caption = sFileName  
 .mnuFileMRU(1).Visible = True  
 .mnuFileBar6.Visible = True

End With

End Sub

**Public Function GetFileNameFromSpec(ByVal sFileSpec As String, sFileName As String) As Integer**

*'Strip the filename and extension from the filespec (drive\path\filename)*

Dim r As Integer

ReDim sList(50) As String

On Error GoTo GetFileNameFromSpec\_Error

If Len(sFileSpec) > 0 Then

    r = ParseDelimString(sFileSpec, "\", sList()) *'delimit all subpaths*  
     sFileName = LCase\$(sList(r)) *'file name is last item in list*

*'something was returned*

If Len(sFileName) > 0 Then GetFileNameFromSpec = True *'return success to caller*

End If

GetFileNameFromSpec\_Exit:

Exit Function

GetFileNameFromSpec\_Error:

Resume 0

Resume GetFileNameFromSpec\_Exit

End Function



## General.bas - ParseDelimString

18

**Public Function ParseDelimString(ByVal sParse As String, ByVal sDelim As String, sFieldStrings() As String) As Integer**

*'Parse "sParse" passed here. Put resulting parsed names in a list called sFieldStrings.  
'Use sDelim as the delimiter to parse string.  
'Trim any leading and trailing spaces from each field.  
'sFieldString list must pre-exist before calling here, and should be big enough to  
'hold all delimited strings.  
'The list contains fields in the order they appeared from left to right.  
'Function returns number of fields found*

*Dim i As Integer 'loop counter  
Dim iDelim1 As Integer, iDelim2 As Integer 'marks beginning and end of a field*

*If Len(sParse) = 0 Then Exit Function 'exit if no chars in string  
iDelim1 = 0 'set first delim marker to beginning of line  
If Right\$(sParse, Len(sDelim)) <> sDelim Then 'see if a delim is already at the end of the string  
sParse = sParse + sDelim 'put a delim at end of line  
End If*

*'Note: an Erase method can not be used as it redims the array to only a few elements  
For i = 0 To UBound(sFieldStrings) 'clear out old data from the array  
sFieldStrings(i) = ""  
Next i*

*i = 0  
Do While iDelim1 < Len(sParse) 'keep looking til all delims are found  
iDelim2 = InStr(iDelim1 + 1, sParse, sDelim) 'look for delim in string  
'get field from string, trim off spaces and put field into list  
sFieldStrings(i + 1) = Trim\$(Mid\$(sParse, iDelim1 + 1, iDelim2 - iDelim1 - 1))  
iDelim1 = iDelim2 'reset first delim marker to latest one found  
i = i + 1 'increment field counter  
Loop  
ParseDelimString = i 'repeat search  
'put parsed items count in element 0 of list*

**End Function**

**Public Function SaveDataToNewFile() As Integer**

*'Get a filename from the common dialog  
'Setup the common dialog control prior to showing it*

*On Error GoTo SaveDataToNewFile\_Error  
With frmMain.dlgCommonDialog  
.Flags = cdiOFNOverwritePrompt Or cdiOFNCreatePrompt Or cdiOFNPathMustExist Or cdiOFNExplorer Or cdiOFNExtensionDifferent  
Or cdiOFNNoReadOnlyReturn Or cdiOFNHideReadOnly  
.CancelError = True 'generate error if CANCEL button is pressed  
.InitDir = App.Path + "\Patient Data"  
.Filter = "CycloTech Data File \*.cpd|\*.cpd"  
.DialogTitle = "Save Patient Data As..."  
.DefaultExt = ".CPD" 'append "Structure" extension when saving.  
If PAT\_DATA.sPatientDataFileName = "" Then  
.filename = PAT\_DATA.sPatientLastName + " " + PAT\_DATA.sPatientFirstName + " " + PAT\_DATA.sPatientID + ".cpd" 'set a  
default file name  
Else  
.filename = PAT\_DATA.sPatientDataFileName  
End If  
.ShowSave 'save as dialog  
End With*

*PAT\_DATA.sPatientDataFileName = frmMain.dlgCommonDialog.filename  
SavePatientData PAT\_DATA.sPatientDataFileName  
SaveDataToNewFile = True*

*SaveDataToNewFile\_Exit:  
Exit Function*

*SaveDataToNewFile\_Error:*

## General.bas - SaveDataToNewFile

19

```

If Err = cdlCancel Then           'cancel button was pressed in dialog
    SaveDataToNewFile = vbCancel
    Resume SaveDataToNewFile_Exit
End If
SaveDataToNewFile = False

```

End Function

### Public Function SavePatientData(ByVal sFileSpec As String) As Integer

*'Save all of the patient data currently in memory to a disk file  
'Return a true if save was successful, false if not, and vbCancel if user cancelled*

```

Dim sTemp As String, r As Integer, l As Integer, sSection As String
Dim lCheckSumTally As Long

```

On Error GoTo SavePatientData\_Error

```

r = GetFileNameFromSpec(frmMain.dlgCommonDialog.filename, sTemp) 'save the dir was selected by user

```

*'We need to confirm with user that it is desired to save these file modifications under the same name as the one that was just loaded.*

```

If Len(frmMain.dlgCommonDialog.filename) And frmMain.dlgCommonDialog.filename = UCase$(Proj.sStructFileName) Then
    MSGS = "You are about to save changes to the same file they were loaded from!"
    MSGS = MSGS + " Are you sure you want to do this?"
    Beep
    r = MsgBox(MSGS, MB_YES_NO, "Confirm Over Write")
    If r = ID_NO Then Exit Function 'oops, user almost made mistake, exit sub
End If

```

*'Now save the data to the file*

```

r = GetFileNameFromSpec(sTemp, sFileSpec) 'hold the name of the file
sFileSpec = App.Path & "\Patient Data"

```

```

frmMain.MousePointer = vbHourglass
DoEvents

```

sSection = "Device Data"

```

SaveINISetting sFileSpec, sSection, "Date Saved To File", Now
SaveINISetting sFileSpec, sSection, "Host Software Version", CStr(App.Major & "." & App.Minor & "." & App.Revision)
SaveINISetting sFileSpec, sSection, "Firmware Version", PAT_DATA.sFirmwareVer
SaveINISetting sFileSpec, sSection, "Last Download Date", CDate(PAT_DATA.dLastDownloadDate) 'short date must be used to prevent
error when loading back
SaveINISetting sFileSpec, sSection, "Device Init Date", CDate(PAT_DATA.iDeviceInitDate)
SaveINISetting sFileSpec, sSection, "Events Ref Date Time", CDate(PAT_DATA.dDeviceRefDateTime)
SaveINISetting sFileSpec, sSection, "Last Name", PAT_DATA.sPatientLastName
SaveINISetting sFileSpec, sSection, "First Name", PAT_DATA.sPatientFirstName
SaveINISetting sFileSpec, sSection, "Serial Number", PAT_DATA.sSerialNumber
SaveINISetting sFileSpec, sSection, "Patient ID", PAT_DATA.sPatientID
SaveINISetting sFileSpec, sSection, "Organ", PAT_DATA.sOrgan
SaveINISetting sFileSpec, sSection, "Organ Reference Number", CStr(GetOrganRefNumber())
SaveINISetting sFileSpec, sSection, "Tx Center", PAT_DATA.sTxCenter
SaveINISetting sFileSpec, sSection, "Drug", PAT_DATA.sDrug
SaveINISetting sFileSpec, sSection, "Drug Reference Number", CStr(GetDrugRefNumber())
SaveINISetting sFileSpec, sSection, "Dose Size", PAT_DATA.sDoseSize
SaveINISetting sFileSpec, sSection, "Doses Per Day", CStr(PAT_DATA.iDosesPerDay)
SaveINISetting sFileSpec, sSection, "Dose Resolution", PAT_DATA.sDoseResolution
SaveINISetting sFileSpec, sSection, "Medication Remaining", PAT_DATA.sMedRemaining
SaveINISetting sFileSpec, sSection, "Battery Change Timer", PAT_DATA.sBatteryChangeTimer
SaveINISetting sFileSpec, sSection, "Lockout Hours Between Doses", PAT_DATA.sDoseLockoutHours

```

```

For i = 1 To 14
    SaveINISetting sFileSpec, sSection, "Patient Score Data " + CStr(i), PAT_DATA.sScoreData(i)
Next i

```

```

For i = 1 To giMaxDoseTimes
    If PAT_DATA.dPrescribedDoseTime(i) >= 0 Then

```

## General.bas - SavePatientData

20

```

SaveINISetting sFileSpec, sSection, "Prescribed Dose Time " + CStr(i), Format$(PAT_DATA.dPrescribedDoseTime(i),
gsTimeDisplayFormat)
Else
SaveINISetting sFileSpec, sSection, "Prescribed Dose Time " + CStr(i), " None"
End If
Next i

```

*'This section is finished. Go compute the checksum and save it.*  
 lChecksumTally = ComputeIniSectionChecksum(sFileSpec, sSection)  
 SaveINISetting sFileSpec, "General", "Device Data Validation", CStr(lChecksumTally)

```

'Before saving new event data, clear out the old strings
sSection = "Event Data"
r = WritePrivateProfileString(sSection, ByVal 0&, ByVal 0&, sFileSpec)
SaveINISetting sFileSpec, sSection, "Event Count", CStr(PAT_DATA.iEventData(0))
For i = 1 To PAT_DATA.iEventData(0) 'total number of events
  sTemp = Format$(PAT_DATA.dEventData(i), "General Date") + ", "
  Select Case PAT_DATA.byteEventType(i)
    Case giEVENT_DOSE_TAKEN
      sTemp = sTemp + "Dose Taken, "
    Case giEVENT_DOSE_CHANGED
      sTemp = sTemp + "Dose Change, "
    Case giEVENT_USER_DEFINED
      sTemp = sTemp + "Custom Event, "
  End Select
  sTemp = sTemp + CStr(PAT_DATA.iEventData(i))
  sTemp = sTemp + ", " + PAT_DATA.sUserData1(i)
  sTemp = sTemp + ", " + PAT_DATA.sUserData2(i)
  sTemp = sTemp + ", " + PAT_DATA.sUserData3(i)
  SaveINISetting sFileSpec, sSection, CStr(i), sTemp
Next i

```

*'This section is finished. Go compute the checksum and save it.*  
 lChecksumTally = ComputeIniSectionChecksum(sFileSpec, sSection)  
 SaveINISetting sFileSpec, "General", "Event Data Validation", CStr(lChecksumTally)

```

sSection = "Device Error Flags"
SaveINISetting sFileSpec, sSection, "Fatal", CStr(PAT_DATA.bErrorFatal)
SaveINISetting sFileSpec, sSection, "Non Fatal", CStr(PAT_DATA.bErrorNonFatal)
SaveINISetting sFileSpec, sSection, "Dose Size", CStr(PAT_DATA.bErrorDoseSize)
SaveINISetting sFileSpec, sSection, "Med Remaining", CStr(PAT_DATA.bErrorMedRemaining)
SaveINISetting sFileSpec, sSection, "Memory Full", CStr(PAT_DATA.bErrorMemoryFull)
SaveINISetting sFileSpec, sSection, "Brownout", CStr(PAT_DATA.bErrorBrownOut)

```

*'This section is finished. Go compute the checksum and save it.*  
 lChecksumTally = ComputeIniSectionChecksum(sFileSpec, sSection)  
 SaveINISetting sFileSpec, "General", "Device Error Flags Validation", CStr(lChecksumTally)

```

gbPatientDataNotSaved = False
r = GetFileNameFromSpec(sFileSpec, PAT_DATA.sPatientDataFileName) 'hold the name of the file
UpdateRecentFileMenu sFileSpec
frmMain.mnuFileSave.Enabled = True
SavePatientData = True

```

```

SavePatientData_Exit:
On Error GoTo 0
frmMain.MousePointer = vbDefault
Exit Function

```

```

SavePatientData_Error:
SavePatientData = False

```

## General.bas - SavePatientData

21

Resume SavePatientData\_Exit *'exit anyway for now*

End Function

**Public Sub PopulateDeviceCommDialog(DataStruct As DeviceDataStruct, SourceForm As Form)**

*'There are two possible dialogs that have the same controls on them.*

*'This common procedure will populate both*

*On Error Resume Next 'not all text boxes will appear on every form*

*Dim i As Integer*

*With SourceForm*

*'Show custom labels from config file if there were any*

*.Label1(3) = gsCustomLblPatientLastName*

*.Label1(1) = gsCustomLblPatientFirstName*

*.Label1(5) = gsCustomLblPatientID*

*.Label1(6) = gsCustomLblTxCenter*

*.Label1(7) = gsCustomLblDrug*

*.Label1(0) = gsCustomLblOrgan*

*..*

*If TypeOf .txtDrug Is SSPanel Then*

*.txtDrug = DataStruct.sDrug*

*ElseIf TypeOf .txtDrug Is ComboBox Then 'it is a list box*

*.txtDrug.Clear*

*For i = 1 To gsDrugNames(0)*

*.txtDrug.AddItem gsDrugNames(i)*

*Next i*

*For i = 0 To .txtDrug.ListCount - 1*

*If .txtDrug.List(i) = DataStruct.sDrug Then*

*.txtDrug.ListIndex = i*

*Exit For*

*End If*

*Next i*

*End If*

*If TypeOf .txtOrgan Is SSPanel Then*

*.txtOrgan = DataStruct.sOrgan*

*ElseIf TypeOf .txtOrgan Is ComboBox Then 'it is a list box*

*.txtOrgan.Clear*

*For i = 1 To gsOrganNames(0)*

*.txtOrgan.AddItem gsOrganNames(i)*

*Next i*

*For i = 0 To .txtOrgan.ListCount - 1*

*If .txtOrgan.List(i) = DataStruct.sOrgan Then*

*.txtOrgan.ListIndex = i*

*Exit For*

*End If*

*Next i*

*End If*

*.txtPatientLastName = DataStruct.sPatientLastName*

*.txtPatientFirstName = DataStruct.sPatientFirstName*

*.txtPatientID = DataStruct.sPatientID*

*.txtTxCenter = DataStruct.sTxCenter*

*.txtSerialNumber = DataStruct.sSerialNumber*

*.txtDoseSize = "" + DataStruct.sDoseSize*

## General.bas - PopulateDeviceCommDia

22

```

If DataStruct.iEventData(0) Then
    .txtEventCount = "" + CStr(DataStruct.iEventData(0)) + ""
Else
    .txtEventCount = ""
End If

If PAT_DATA.dLastDownloadDate Then
    .txtLastRetrievalDate = "" + Format$(CDate(DataStruct.dLastDownloadDate), "Short Date") + " " + Format$(CDate(DataStruct.
dLastDownloadDate), "Medium Time") + ""
Else
    .txtLastRetrievalDate = ""
End If

.txtPatientLastName 'take focus away from list box after it was set

For i = 1 To giMaxDoseTimes
    If DataStruct.dPrescribedDoseTime(i) >= 0 Then
        .txtDoseTime(i) = "" + Format$(DataStruct.dPrescribedDoseTime(i), gsTimeDisplayFormat) + ""
    End If
Next i

.txtDosesPerDay = "" + CStr(DataStruct.iDosesPerDay) + ""
.txtDoseResolution = "" + DataStruct.sDoseResolution + ""
.txtDoseLockoutHours = "" + DataStruct.sDoseLockoutHours + ""
.txtMedicationRemaining = "" + DataStruct.sMedRemaining + ""
If DataStruct.iDeviceInitDate Then
    .txtDeviceStarted = "" + Format$(CDate(DataStruct.iDeviceInitDate), "Medium Date")
End If
.txtBatteryChangeTimer = "" + DataStruct.sBatteryChangeTimer + ""

'set indicators for error flags
With DataStruct
    If .bErrorFatal Or .bErrorNonFatal Or .bErrorDoseSize Or .bErrorMedRemaining Or .bErrorMemoryFull Or .bErrorBrownOut Then
        SourceForm.imgErrorReceived.Visible = True 'errors were found
        SourceForm.lblErrorsReceived.Visible = True
    Else
        SourceForm.imgErrorsReceived.Visible = False 'no errors exist
        SourceForm.lblErrorsReceived.Visible = False
    End If
End With

End With
On Error GoTo 0

End Sub

```

## Public Sub SaveProgramPreferences()

```

Dim i As Integer, sSection As String, sFileSpec As String
sSection = "Preferences"
SaveINISetting gsAppIniFileSpec, sSection, "Date Display Format", gsDateDisplayFormat
SaveINISetting gsAppIniFileSpec, sSection, "Time Display Format", gsTimeDisplayFormat
SaveINISetting gsAppIniFileSpec, sSection, "Compliance Time Range", CStr(gsngComplianceTimeRange)

'Save the names of the most recently used files from the menu
For i = 1 To frmMain.mnuFileMRU.UBound
    SaveINISetting gsAppIniFileSpec, "Recent Files", CStr(i), frmMain.mnuFileMRU(i).Caption
Next i

SaveINISetting gsAppIniFileSpec, "Options", "Current Tip", CStr(giCurrentTip)

'Save Settings of Calendar Form
SaveINISetting gsAppIniFileSpec, "Calendar Settings", "chkDosesMissed", CStr(CAL_DEFAULTS.chkDosesMissed)
SaveINISetting gsAppIniFileSpec, "Calendar Settings", "chkDosesNotComplied", CStr(CAL_DEFAULTS.chkDosesNotComplied)
SaveINISetting gsAppIniFileSpec, "Calendar Settings", "chkDosesTaken", CStr(CAL_DEFAULTS.chkDosesTaken)
SaveINISetting gsAppIniFileSpec, "Calendar Settings", "chkDoseChanged", CStr(CAL_DEFAULTS.chkDoseChanged)

```

## General.bas - SaveProgramPreference

'Save Settings of Patient Summary Form

SaveINISetting gsAppIniFileSpec, "Patient Summary Settings", "cmboDataToView", CStr(PAT\_SUM\_DEFAULTS.cmboDataToView)  
 SaveINISetting gsAppIniFileSpec, "Patient Summary Settings", "cmboChartType", CStr(PAT\_SUM\_DEFAULTS.cmboChartType)

End Sub

**Public Function FindFirstMatchingDateInArray(DataStruct As DeviceDataStruct, ByVal IBeginDate As Long)**  
*'Find the earliest event date in the global structure that starts on the same day*  
*'as the date passed here. Return 0 if not found or return the index to the date*  
*'if one is found.*

*'Note that this date is not necessarily a dosing event date. It could be any kind of event*

*'Conduct a successive approximation lookup of the date in the array*

Dim i As Integer, iLowIndex As Integer, iHighIndex As Integer, iTestIndex As Integer

iLowIndex = 1 *'start at begin of array*

iHighIndex = DataStruct.iEventData(0) *'stop at end of array*

iTestIndex = (iHighIndex + iLowIndex) / 2

For i = 1 To 7 *'this number of tries is all that is necessary to find the date*

If IBeginDate < Int(DataStruct.dEventDate(iTestIndex)) Then  
 iHighIndex = iTestIndex

ElseIf IBeginDate > Int(DataStruct.dEventDate(iTestIndex)) Then  
 iLowIndex = iTestIndex

ElseIf IBeginDate = Int(DataStruct.dEventDate(iTestIndex)) Then  
 iHighIndex = iTestIndex  
 FindFirstMatchingDateInArray = iTestIndex

End If

iTestIndex = (iHighIndex + iLowIndex + 0.5) / 2

Next i

End Function

**Public Function FindClosestDateInArray(DataStruct As DeviceDataStruct, ByVal IFromDate As Long) As Long**  
*'Find the latest event date in the global structure that starts on the same day*  
*'as the date passed here. Return 0 if not found or return the index to the date*  
*'if one is found.*

*'If a 0 value is passed here then find the most recent date in the array.*

*'Note that this date is not necessarily a dosing event date. There is a separate*

*'procedure to find that date.*

*'rgh if necessary for faster speed, this procedure can be recoded to do a successive approximation*

Dim i As Integer

If IFromDate = 0 Then IFromDate = 99999

For i = 1 To DataStruct.iEventData(0) *'find the latest date in the array*

If IFromDate <= Int(DataStruct.dEventDate(i)) Then

FindClosestDateInArray = i

Exit For

End If

Next i

End Function

## General.bas - SaveINISetting

24

```

Public Sub SaveINISetting(ByVal sFileName As String, ByVal sSection As String, ByVal sKeyField As String, s
    Dim l As Long
    l = WritePrivateProfileString(sSection, sKeyField, sValue, sFileName)
End Sub

```

```

Public Function ValidateDoseNumbers(frmTarget As Form)

```

*'Ensure that there are at least as many dose times as there are  
'for the number of doses per day.*

```

    Dim i As Integer, iDailyDoseCounts As Integer, iDosesPerDay As Integer
    If Len(PAT_DATA.sPatientDataFileName) = 0 And Len(PAT_DATA.sSerialNumber) = 0 Then
        ValidateDoseNumbers = True
        Exit Function
    End If

```

```

    With frmTarget
        iDosesPerDay = Val(.txtDosesPerDay)

```

```

        For i = 1 To 4
            If IsDate(.txtDoseTime(i)) Then
                iDailyDoseCounts = iDailyDoseCounts + 1
            End If
        Next i

```

```

        If iDailyDoseCounts = iDosesPerDay Then
            ValidateDoseNumbers = True

```

```

        Else
            Beep
            MsgBox "You have indicated " + CStr(iDosesPerDay) + " Doses Per Day, yet " + CStr(iDailyDoseCounts) + " Dose Times were
            entered. They must match.", vbExclamation, "Mis-matched Dosing Values"
            .txtDosesPerDay.SetFocus
            .UpDownDoseTime(4).SetFocus
        End If

```

```

    End With

```

```

End Function

```

```

Public Function ValidatePatientDataSaved()

```

*'Ensure that the patient data in memory is saved before proceeding to load new data from device  
'Return true if successful, else return vbCancel if user cancelled*

```

    Dim r As Integer

```

```

    ValidatePatientDataSaved = True    'this is the default condition unless set otherwise below
    If gbPatientDataNotSaved Then

```

```

        Beep
        r = MsgBox("The patient data currently in memory has not been saved. Do you want to save it?", vbYesNoCancel + vbQuestion, "
        Patient Data Not Saved")

```

```

        If r = vbYes Then
            r = SaveDataToNewFile()
            If r = vbCancel Then ValidatePatientDataSaved = vbCancel    'cancelled from the save as dialog

```

```

        ElseIf r = vbCancel Then
            ValidatePatientDataSaved = vbCancel    'cancelled from message box

```

```

        End If

```

```

    End If

```

---

General.bas - ValidatePatientDataSaved

---

25

End Function

---

**Public Sub Wait(ByVal dSeconds As Double)***'wait for the specified time passed here. then return to caller.*

Dim dDelay As Double

*dDelay = DateAdd("s", dSeconds, CDbl(Now))**dDelay = CDbl(Now) + (dSeconds / 86400) '36400 seconds in a day*

Do

DoEvents

DoEvents

Loop While (dDelay &gt;= CDbl(Now))

End Sub



## Comm.bas - File Declarations

26

```

Attribute VB_Name = "modComm"
Option Explicit
'Global definitions for device communication.
'by Glen Hamilton 10/5/97

' for RS232 communication
Public gbCommTimerExpired As Integer 'this flag is set when the comm timer expires
Public giCommPort As Integer 'Communication Port #
Public gsCommDeviceSettings As String 'speed settings (ie 2400.N.8.2)
Public gbCommReplyPending As Boolean 'a command was just sent, and reply is pending
Public gbCommBusy As Boolean 'a command is in progress. Get's cleared when reply is received or times out
Public gbCommOK As Integer 'needs to be an integer (no boolean) keeps current status of communications. false= no
comm, true = comm ok, any other value for simulation
Public giDeviceResponseWait As Integer 'milisecs to wait for next char before assuming end of received string
Private Const ERR_COMM_BADRESPONSE = 31001
Private Const ERR_COMM_TIMEOUT = 30998
Private Const ERR_COMM_STRINGLENGTH = 30997
Private Const ERR_COMM_BUSY = 30996
Private Const ERR_COMM_CHECKSUM = 30995

Public Const ERR_DATA_CHECKSUM = 99997
Public Const ERR_NEWER_HOST_SOFTWARE = 99998 'set when device returns custom data that was saved with a newer revision
level

'Device communications
Public gbKeepPollingDevice As Boolean 'when true, continuous polling of device is done

'Define some application specific variables & constants
Public gsAppIniFileSpec As String

Type DeviceDataStruct
sPatientLastName As String '(16 bytes) uses 1st 16 byte block of the patient/pharmacy ID & Names
sPatientFirstName As String '(16 bytes) uses 1st 16 byte block of the patient/pharmacy ID & Names
sPatientID As String
sDrug As String '(16 bytes) uses 2nd 16 byte block of the patient/pharmacy ID & Names
sOrgan As String '(16 bytes) uses 3rd 16 byte block of the patient/pharmacy ID & Names
sTxCenter As String '(16 bytes) uses 3rd 16 byte block of the patient/pharmacy ID & Names
sSerialNumber As String '(10 bytes) device serial number
sFirmwareVer As String 'Rev version and date of firmware
sDoseSize As String '(1 byte) stored here in mg. The device uses "ml" (100mg = 1 ml)
'Device Dose size is in optical ticks (0 to 200) max dose = 5 ml.

sPatientDataFileName As String 'file path and filename of the data in memory

'Note: the daily prescribed dosing times below are stored in fractional days. This is done
'to speed display operations and reduce the amount of memory needed. The device actually stores
'these values as intervals relative to 1:00 in the morning. Thus, the times are converted to
'intervals when communicating with the device.
dPrescribedDoseTime(4) As Double 'doses due during the day (prescribed) usually a max of four
IDosesPerDay As Integer '(1 byte) # of doses per day (1 to 4)
sDoseResolution As String '(1 byte) Called "Dose Conversion" in firmware.
'Optical ticks to mg multiplier. (1E 2 ticks = 10 mg.)
'Optical ticks are fixed at 0.05 ml per tick.
sMedRemaining As String '(2 bytes) Medication "Supply volume" remaining (in optical ticks)
sScoreData(14) As String 'Today's score(14 bytes for all scores) of last 14 days doses taken. Circular buffer.
'valid data is value from 0-4 representing number of doses taken each day.
'Note: The "Score pointer" points to the current day.

'Note that the following arrays can not be larger than 1500 events or else the space limit
'of 64K will be exceeded. If necessary in the future to have more events than this for
'a single file then make a separate array for the diagnostic data or temp data.
iEventData(1400) As Integer 'the data occurring for the event data. Might be a dose size, error flags, etc.
dEventDate(1400) As Double 'list of dose days in order of first taken to most recent
byteEventType(1400) As Byte 'value =0 if it is a dose taken
'value = 1 if data event is a dose command change
'value = 2 if user entered entered
sUserData1(1400) As String 'user entered data in the first column of the gnd
sUserData2(1400) As String 'user entered data in the first column of the gnd

```

## Comm.bas - File Declarations

27

```

sUserData3(1400) As String      'user entered data in the first column of the grid

sClock As String                '(2 bytes) 10 minute resolution "0000" = 1 am on first dose day.
IDeviceInitDate As Long         'date the device started
dDeviceRefDateTime As Double    'date and time that all events are referenced to.
sBatteryChangeTimer As String   '(2 bytes) Battery change timer, in 10 minute increments
sDoseLockoutHours As String     '(1 byte) Hours to lockout dosing after a dose is taken
bErrorFatal As Boolean          'true if this flag was set in the returned flags string
bErrorNonFatal As Boolean       'true if this flag was set in the returned flags string
bErrorDoseSize As Boolean       'true if this flag was set in the returned flags string
bErrorMedRemaining As Boolean   'true if this flag was set in the returned flags string
bErrorMemoryFull As Boolean     'true if this flag was set in the returned flags string
bErrorBrownOut As Boolean       'true if this flag was set in the returned flags string
bErrorsExist As Boolean         '(1 byte) Bits are set if various errors have occurred and have not
                                'been corrected. A value of "0" is normal (no errors). Errors
                                'are corrected by either correcting the specific situation or
                                'resetting & reloading the dosing parameters.
                                'B0=1 if fatal system failure
                                'B1=1 if non-fatal system failure has occurred
                                'B2=1 if error has occurred in Dose Size Volume
                                'B3=1 if error has occurred in Supply Volume value
                                'B4=1 if compliance memory is near full
                                'B5=1 if brownout (low voltage) occurred

dLastDownloadDate As Double     'date of last data retrieval from device

End Type
Public PAT_DATA As DeviceDataStruct
Public TEMP_DATA As DeviceDataStruct

Public gsDrugNames(25) As String 'names of drugs used to populate the list boxes on dialogs
Public gsOrganNames(25) As String 'names of gsOrganNames used to populate the list boxes on dialogs

Public Const giEVENT_DOSE_TAKEN = 0
Public Const giEVENT_DOSE_CHANGED = 1
Public Const giEVENT_USER_DEFINED = 2

'These values indicate the string position (returned from the device) where each element
'begins. This is the string that is returned when a request for "all memory" is sent.
'See above structures for more detail information about format.
Public Const DATA_BEGIN_DOSE_SIZE = 1 '1 byte
Public Const DATA_BEGIN_DOSE_INTERVAL1 = 1 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSE_INTERVAL2 = 2 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSE_INTERVAL3 = 3 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSE_INTERVAL4 = 4 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSES_PER_DAY = 5 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSE_CONVERSION = 6 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSE_LOCKOUT_HOURS = 7 * 2 + 1 '1 byte
Public Const DATA_BEGIN_DOSE_SCORE_DAY_POINTER = 8 * 2 + 1 '1 byte
Public Const DATA_BEGIN_MED_REMAINING = 9 * 2 + 1 '2 bytes
Public Const DATA_BEGIN_CLOCK = 11 * 2 + 1 '2 bytes clock starts at 1am on first dosing day (10 min increments)
Public Const DATA_BEGIN_BATTERY_CHANGE_TIMER = 13 * 2 + 1 '2 bytes
Public Const DATA_BEGIN_ERROR_FLAGS = 15 * 2 + 1 '1 byte
Public Const DATA_BEGIN_PREV_DOSE_PARAMS = 16 * 2 + 1 '16 bytes of copy of prev dosing params
                                '(used for error checking internal to dispenser)
Public Const DATA_BEGIN_KEY_BITS = 32 * 2 + 1 'activation of keys on device
Public Const DATA_BEGIN_LIFE_COUNT = 33 * 2 + 1 'LSB in 33, MSB in 34
Public Const DATA_BEGIN_LIFE_COMPLETION = 35 * 2 + 1 '=0 when life cycle is programmed, =1 when life test completes
                                'successfully
Public Const DATA_BEGIN_COMPENSATION_FACTOR = 36 * 2 + 1 'values from 64-192 (128= 1.0 factor)
Public Const DATA_BEGIN_SERIAL_NUMBER = 38 * 2 + 1 '10 bytes
Public Const DATA_BEGIN_CUSTOM1 = 48 * 2 + 1 '16 bytes of patient/pharmacy ID & names
Public Const DATA_BEGIN_CUSTOM2 = 64 * 2 + 1 '16 bytes of patient/pharmacy ID & names
Public Const DATA_BEGIN_CUSTOM3 = 80 * 2 + 1 '16 bytes of patient/pharmacy ID & names
Public Const DATA_BEGIN_CUSTOM4 = 96 * 2 + 1 '16 bytes of patient/pharmacy ID & names
Public Const DATA_BEGIN_SCORE = 112 * 2 + 1 '14 bytes
Public Const DATA_BEGIN_COMPLIANCE_CHECKSUM = 128 * 2 + 1 '2 bytes Includes compliance pointer and data
                                '(up to data word 1 before data pointed to by comp pointer

```

## Comm.bas - File Declarations

28

```

Public Const DATA_BEGIN_COMPLIANCE_POINTER = 130 * 2 + 1    '2 bytes points to next location after end of current compliance
data
    'base value = 132 (0x0084)
Public Const DATA_BEGIN_COMPLIANCE_DATA = 132 * 2 + 1    '-1900 bytes max. Array of 2 byte values for dose compliance
history.
    'Clock time values (in 10 minutes resolution from start) when each dose
    'was taken. Represented by values 0-65279 (0-0x0000). Each dose time
    'is changed via the "Set Mode", a value between 0x0000 and 0x0000 is written
    'with the LSD byte representing the dose size. When compliance memory is
    'cleared, the current dose size is always written as the first location in
    'the compliance memory.

```

**Public Sub ChangeBatteriesRequest()**

```

Dim r As Integer, IErrorCode As Long, sMSG As String

```

```

sMSG = "You should continue only if you are replacing the batteries in the device."
sMSG = sMSG + " This ensures that the battery time counter will be accurate." + vbCrLf + vbCrLf
sMSG = sMSG + " Did you just replace the batteries or are you about to change them now?"
r = MsgBox(sMSG, vbQuestion + vbYesNo + vbDefaultButton2, "Change Batteries")

```

```

If r = vbNo Then Exit Sub

```

```

gbKeepPollingDevice = False    'stop polling for now
Wait 0.25

```

```

On Error GoTo ChangeBatteriesRequest_Error

```

```

r = Comm_SendResetClockAndBattery(IErrorCode)

```

```

If IErrorCode Then

```

```

    Error IErrorCode    'error number

```

```

Else

```

```

    sMSG = "Replace the device batteries now and retrieve data from the device again when complete."

```

```

    r = MsgBox(sMSG, vbExclamation, "Change Batteries")

```

```

End If

```

```

ChangeBatteriesRequest_Exit:

```

```

gbKeepPollingDevice = True    'continue polling device
Exit Sub

```

```

ChangeBatteriesRequest_Error:

```

```

DisplayErrorMessage IErrorCode

```

```

Resume ChangeBatteriesRequest_Exit

```

```

End Sub

```

**Public Function Comm\_CheckComm(IErrorCode As Long) As Integer**

```

'Check the device communication by sending a command and waiting for a reply.

```

```

'If no reply is received, then return a "false" flag to caller.

```

```

'Important Note: Due to the way the firmware was designed for the device, it seems not
'to return anything if the command is in error. This is not good because
'we would not know whether or not a failed reply is due to a back cable, incorrect comm port
'or settings, etc. Hopefully in a future version, the comm check can return some sort of
'character to indicate that a common byte was received, but could not be interpreted correctly.

```

```

Dim sOut As String, sChecksum As String, sin As String

```

```

sOut = "Pp"

```

```

    'this is the code for checking communication with device

```

```

CreateChecksum sOut, sChecksum    'calculate a checksum

```

```

sOut = sOut + sChecksum + "!"    'append checksum and ending string identifier

```

```

If Not frmMain.CommDevice.PortOpen Then frmMain.CommDevice.PortOpen = True

```

```

gbCommBusy = True

```

```

    'prevent other procedures from communicating with device

```

```

frmMain.CommDevice.InputLen = 0

```

```

    'clear input buffer

```

## Comm.bas - Comm\_CheckComm

29

```

frmMain.CommDevice.Output = sOut      'send string to device
gbCommReplyPending = True             'prevent other procedures from communicating with device
SetCommTimer giDeviceResponseWait    'set timer to wait for response

IErrorCode = 0                        'reset error code
Do
    If gbCommTimerExpired Then         'timer event sets this to true
        IErrorCode = ERR_COMM_TIMEOUT 'no response, get the error code
        GoTo Comm_CheckComm_Exit      'return to calling procedure
    End If
    DoEvents
Loop Until frmMain.CommDevice.InBufferCount > 0 'loop till a reply is received or timeout occurs

sin = frmMain.CommDevice.Input        'Read response from serial port
'comm is ok
If sin = "$" Then Comm_CheckComm = True 'return success to caller

Comm_CheckComm_Exit:
If frmMain.CommDevice.PortOpen Then frmMain.CommDevice.PortOpen = False 'Close the serial port
gbCommReplyPending = False          'reset flag
gbCommBusy = False                  'reset flag

End Function

```

### Public Function Comm\_GetDeviceReply(sReply As String, IErrorCode As Long) As Integer

'A command should have been just sent to the device from another procedure and a reply is pending  
 'Get the reply into 'sReply' and return to caller.  
 'Return false if no reply and set IErrorCode to reason.  
 'Return ERR\_COMM\_TIMEOUT if no reply from the device.  
 'error code = 0 if comm is already busy.  
 'If reply, then return number of characters received.  
 'Close comm port once a reply is received or if an error occurs.

Dim iLastBufferCount As Integer, r As Integer

On Error GoTo Comm\_GetDeviceReply\_Error  
 gbCommReplyPending = True 'set busy flag  
 frmMain.MousePointer = vbHourglass

'Open comm port in case it is closed  
 'prevent device unavailable error

If frmMain.CommDevice.PortOpen = False Then frmMain.CommDevice.PortOpen = True 'open port  
 sReply = "" 'init reply

'Wait for first char to arrive

SetCommTimer giDeviceResponseWait '20 milliseconds is normally sufficient  
 Do Until frmMain.CommDevice.InBufferCount > 0

DoEvents

'timer event sets this to true

If gbCommTimerExpired Then Error ERR\_COMM\_TIMEOUT 'return message to caller. No response  
 Loop

'First char has been received

'Wait for all data to arrive

iLastBufferCount = -1 'init buffer count

Do While frmMain.CommDevice.InBufferCount > iLastBufferCount 'characters are still arriving  
 iLastBufferCount = frmMain.CommDevice.InBufferCount 'remember intermediate count

SetCommTimer giDeviceResponseWait 'this value works as low as 25 milliseconds  
 Do Until gbCommTimerExpired = True 'wait for timer to expire

DoEvents

DoEvents

Loop

Loop 'loop if characters are still coming in

'All data has arrived or time has been too long

## Comm.bas - Comm\_GetDeviceReply

30

```

r = frmMain.CommDevice.InBufferCount      'get character length
sReply = frmMain.CommDevice.Input         'read string from buffer
Comm_GetDeviceReply = Len(sReply)         'return buffer length to caller (- checksum)

Comm_GetDeviceReply_Exit:
'prevent device unavailable error
If frmMain.CommDevice.PortOpen = True Then frmMain.CommDevice.PortOpen = False      'close port if open
frmMain.MousePointer = vbDefault
On Error GoTo 0                          'clear error status
gbCommReplyPending = False               'reset pending flag
gbCommBusy = False                       'reset busy flag
Exit Function

Comm_GetDeviceReply_Error:
IErrorCode = Err                        'return error code to caller
"Resume 0 'for testing only
Resume Comm_GetDeviceReply_Exit
End Function

```

**Public Function Comm\_ReadFirmwareVersion(DataStruct As DeviceDataStruct, IReturnError As Long) As Integer**  
 Dim sOut As String, sChecksum As String, sin As String, IErrorCode As Long, r As String

```

sOut = "W"                               'this is the code for version number
CreateChecksum sOut, sChecksum           'calculate a checksum
sOut = sOut + sChecksum + "I"           'append checksum and ending string identifier

```

```

If Not frmMain.CommDevice.PortOpen Then frmMain.CommDevice.PortOpen = True

```

```

frmMain.CommDevice.InputLen = 0          'clear input buffer
frmMain.CommDevice.Output = sOut         'send string to device
r = Comm_GetDeviceReply(sin, IErrorCode)

```

```

If IErrorCode = 0 Then                   'comm was received
  r = ValidateChecksum(sin)
  If r Then
    DataStruct.sFirmwareVer = Left$(sin, Len(sin) - 5) 'put string in global array
    Comm_ReadFirmwareVersion = True               'return success to caller
  End If

```

```

Else
  IReturnError = IErrorCode
  " DisplayErrorMessage IErrorCode
End If
End Function

```

**Public Sub DisplayCommError(SourceForm As Form)**

```

gbCommOK = False
SourceForm.imgCommStatus.Picture = SourceForm.imgRedLight
SourceForm.lblCommStatus = "No Device Found"

```

```

'Play disconnect sound and show status visually
'Set properties needed by MCI to open
With SourceForm.MMControl1
  .Notify = False
  .Wait = False
  .Shareable = False
  .filename = App.Path + "\ProbDetectVoice.wav"

```

```

'Open the MCI WaveAudio device
.Command = "Open"
.Command = "sound"
.Command = "close"
End With

```

## Comm.bas - DisplayCommError

31

End Sub

**Public Sub DisplayCommOk(SourceForm As Form)**

*'Play connect sound and show status visually*  
*'Set properties needed by MCI to open*

gbCommOK = True

With SourceForm.MMControl1

.Notify = False

.Wait = False

.Shareable = False

.filename = App.Path + "\morsecode.wav"

*'Open the MCI WaveAudio device*

.Command = "Open"

.Command = "sound"

.Command = "sound"

.Command = "close"

End With

SourceForm.imgCommStatus.Picture = SourceForm.imgGreenLight

SourceForm.lblCommStatus = "Device Ready"

End Sub

**Public Sub DisplayErrorMessage(ErrorCode As Long)**

Dim sMSG As String

Select Case ErrorCode

Case ERR\_COMM\_TIMEOUT

sMSG = "No response was received from the device to the command just issued. Remove the device from the communicator and re-insert it to ensure that it is seated properly."

Case ERR\_COMM\_CHECKSUM

sMSG = "Data retrieved from the device is corrupted. This probably occurred during transmission. Please read the device again."

Case ERR\_COMM\_BADRESPONSE

sMSG = "The device did not interpret the command properly."

Case ERR\_NEWER\_HOST\_SOFTWARE

sMSG = "The device was previously programmed with a newer version of this software. The data can not be retrieved." + vbCrLf + "Please obtain an updated version this software."

Case Else

sMSG = "An error was detected while communicating with the device. Please try again." + vbCrLf + vbCrLf + Error\$(ErrorCode)

End Select

MsgBox sMSG, , App.ProductName + " Comm Error - " + CStr(ErrorCode)

End Sub

## Comm.bas - GetDrugRefNumber

32

**Public Function GetDrugRefNumber() As Integer***'Find the index to the organ name being using in the global structure*

Dim i As Integer

For i = 1 To UBound(gsDrugNames)

If LCase(PAT\_DATA.sDrug) = LCase(gsDrugNames(i)) Then Exit For

Next i

GetDrugRefNumber = i *'return ref number to caller*

End Function

**Public Function GetOrganRefNumber() As Integer***'Find the index to the organ name being using in the global structure*

Dim i As Integer

For i = 1 To UBound(gsOrganNames)

If LCase(PAT\_DATA.sOrgan) = LCase(gsOrganNames(i)) Then Exit For

Next i

GetOrganRefNumber = i *'return ref number to caller*

End Function

**Private Function InterpretDosingData(DataStruct As DeviceDataStruct, ByVal sData As String, ByVal dChecksumTally As Double, ByVal IChecksum As Long, IErrorCode As Long) As Integer***'Parse apart the dosing data that is passed here and put into global structure.**'Each dosing event is 2 bytes in length.**'The checksum is passed here for comparison to the string.**'The checksum includes the pointer bytes which is why it is passed in.*

Dim sTemp As String, iLowByte As Integer, iHiByte As Integer, iCurrentDoseAmount As Long

Dim r As Integer, iposition As Integer, ICount As Integer, ITemp As Long

On Error GoTo InterpretDosingData\_Error

For iposition = 1 To Len(sData) Step 4 *'there are 2 hex bytes to every dose event*

ICount = ICount + 1

iLowByte = CInt("&H" + Mid\$(sData, iposition, 2)) *'get first of the 2 byte hex value*dChecksumTally = dChecksumTally + iLowByte *'add to checksum*iHiByte = CInt("&H" + Mid\$(sData, iposition + 2, 2)) *'get second of the 2 byte hex value*dChecksumTally = dChecksumTally + iHiByte *'add to checksum*If iHiByte = 255 Then *'indicates a dose change*

DataStruct.byteEventType(iCount) = giEVENT\_DOSE\_CHANGED

iCurrentDoseAmount = CLng(iLowByte) / 40 \* 100 *'convert from ml to mg: this is a new dose size change*

DataStruct.iEventData(iCount) = iCurrentDoseAmount

DataStruct.dEventData(iCount) = CLng(DataStruct.dEventData(iCount - 1)) *'get date from last dose**'This is the very first dose change*

If DataStruct.dEventData(iCount) = 0 Then

DataStruct.dEventData(iCount) = 1

End If

Else *'this is a dose*

DataStruct.byteEventType(iCount) = giEVENT\_DOSE\_TAKEN

DataStruct.iEventData(iCount) = iCurrentDoseAmount *'convert from ml to mg: this is a new dose size change*ITemp = CLng(iHiByte) \* 256 + iLowByte *'date and time is 10 minute intervals since first dose day*

DataStruct.dEventData(iCount) = DateAdd("n", ITemp \* 10, DataStruct.dDeviceRefDateTime)

End If

Next iposition

DataStruct.iEventData(0) = ICount *'put the total number of events in the 0 element of list*

dChecksumTally = dChecksumTally Mod 65536

If dChecksumTally = IChecksum Then

## Comm.bas - InterpretDosingData

33

```

    InterpretDosingData = True    'return success to caller
Else
    IErrorCode = ERR_COMM_CHECKSUM
End If

```

```

InterpretDosingData_Exit:
    On Error GoTo 0
    Exit Function

```

```

InterpretDosingData_Error:
    IErrorCode = Err
    Resume InterpretDosingData_Exit:

```

```

End Function

```

### Public Function ValidateChecksum(ByVal sData As String) As Integer

*'Look at the data string passed here and get the checksum from the  
'end of the string.  
'sDATA should be a string that was returned from the device.  
'The last char in the string is a termination char preceeded by 4 bytes of checksum.*

```

Dim sTemp As String, iByte As Integer, r As Integer, iposition As Integer
Dim IChecksum As Long, IChecksumTally As Long
On Error GoTo ValidateChecksum_Error

```

```

r = Len(sData)
For iposition = 1 To r - 5
    iByte = Asc(Mid$(sData, iposition, 1))
    'Byte = CInt("&H" + Mid$(sData, iposition, 2))
    IChecksumTally = IChecksumTally + iByte    'add to checksum
Next iposition

```

```

ICheckSumTally = IChecksumTally Mod 65536

```

```

sTemp = "&H" + "0" + Mid$(sData, r - 2, 2) + Mid$(sData, r - 4, 2)
ICheckSum = CLng(sTemp)
If IChecksumTally = IChecksum Then ValidateChecksum = True    'pass success flag back to caller

```

```

ValidateChecksum_Exit:
    On Error GoTo 0
    Exit Function

```

```

ValidateChecksum_Error:
    Resume ValidateChecksum_Exit

```

```

End Function

```

### Private Sub InterpretErrorFlags(DataStruct As DeviceDataStruct, ByVal iFlagsByte As Integer)

*'Break out the bits of the flags bytes passed here.  
'Put the results into the global arrays*

```

'if any flags exist, then set this to true
If iFlagsByte Then DataStruct.bErrorsExist = True

```

```

'Parse out flags separately
DataStruct.bErrorFatal = (iFlagsByte And 2)
DataStruct.bErrorNonFatal = (iFlagsByte And 4)
DataStruct.bErrorDoseSize = (iFlagsByte And 8)
DataStruct.bErrorMedRemaining = (iFlagsByte And 16)
DataStruct.bErrorMemoryFull = (iFlagsByte And 32)
DataStruct.bErrorBrownOut = (iFlagsByte And 64)
'remaining upper 3 bits not used at present

```

```

End Sub

```



Comm.bas - Comm\_ReadEntireMemoryContents

34

**Public Function Comm\_ReadEntireMemoryContents(DataStruct As DeviceDataStruct, IReturnError As Long) As String**  
 Dim sOut As String, sChecksum As String, sin As String, IErrorCode As Long, r As String

On Error GoTo Comm\_ReadEntireMemoryContents\_Error

EraseDataInMemory DataStruct

sOut = "Rr" *'this is the code for reading entire memory*

CreateChecksum sOut, sChecksum *'calculate a checksum*

sOut = sOut + sChecksum + "!" *'append checksum and ending string identifier*

If Not frmMain.CommDevice.PortOpen Then frmMain.CommDevice.PortOpen = True

frmMain.CommDevice.InputLen = 0 *'clear input buffer*

frmMain.CommDevice.Output = sOut *'send string to device*

r = Comm\_GetDeviceReply(sin, IErrorCode)

If r Then *'comm was received. Should be at least this many bytes*

r = ValidateChecksum(sin)

If r = False Then

IReturnError = ERR\_COMM\_CHECKSUM

Exit Function

End If

r = ParseMemoryContents(DataStruct, sin, IErrorCode) *'parse out the string*

If IErrorCode Then

IReturnError = IErrorCode

Exit Function

End If

Comm\_ReadEntireMemoryContents = True *'return success to caller*

DataStruct.dLastDownloadDate = Now

gbPatientDataNotSaved = True *'set this flag to true*

Else

IReturnError = IErrorCode

Exit Function

End If

r = Comm\_ReadFirmwareVersion(DataStruct, IErrorCode)

If IErrorCode Then

IReturnError = IErrorCode

End If

Comm\_ReadEntireMemoryContents\_Exit:

On Error GoTo 0

Exit Function

Comm\_ReadEntireMemoryContents\_Error:

IReturnError = Err

Resume Comm\_ReadEntireMemoryContents\_Exit

End Function

Comm.bas - Comm\_SendResetClockAndBattery

35

**Public Function Comm\_SendResetClockAndBattery(IReturnError As Long)**

*'Resets the device clock to an offset that represents 1:00am  
'and resets the battery timer to zero.*

Dim sData As String, sOut As String, sReply As String, sChecksum As String  
Dim r As Integer, IErrorCode As Long, iTemp As Integer

iTemp = CInt(Format(Now, "hh"))  
If iTemp = 0 Then iTemp = 24 *'midnight*  
iTemp = (iTemp - 1) \* 6 *'calc number of 10-minute period \* hours*  
iTemp = iTemp + CInt((Format\$(Now, "nn") - 5) / 10) *'calc number of 10-min periods in this hour*  
sData = CStr(Hex(iTemp))  
If Len(sData) < 2 Then sData = "0" + sData *'ensure string is always 2 bytes long*

sOut = "Kk" + sData *'add data string to command*  
CreateChecksum sOut, sChecksum *'calculate a checksum*  
sOut = sOut + sChecksum + "!" *'append checksum and ending string identifier*  
Comm\_SendDataToDevice(sOut) *'send string to comm port*  
r = Comm\_GetDeviceReply(sReply, IErrorCode)  
If sReply = "\$" Then *'string was successfully interpreted by device*  
Comm\_SendResetClockAndBattery = True *'return success to caller*  
ElseIf sReply = "?" Then *'string was not interpreted properly*  
IReturnError = ERR\_COMM\_BADRESPONSE  
Else  
IReturnError = IErrorCode  
End If

End Function

**Public Function Comm\_SendCustomData(DataStruct As DeviceDataStruct, ByVal sLocation As String, IReturn**

*'There are 4 locations in the device, each containing a 16 byte string.  
'There first location is usually reserved for Patient ID.  
'Any string can be contained in any location.  
'Data is taken from the global structure*

Dim sData As String, sOut As String, sReply As String, sChecksum As String  
Dim r As Integer, IErrorCode As Long, sCustomData As String  
Dim i As Integer, sTemp As String

*'Determine the appropriate command for the location of the data to be stored.  
'There are 4 fields in the device containing 16 characters each. In the original  
'device design, this was intended to contain 4 separate pieces of information.  
'The client has now decided that some fields are too short and others are too long.  
'Thus, the fields are combined to be one string of 64 characters.*

*'Data Structure Rev level = giLEN\_REV\_DATA\_STRUCTURE*  
*'Patient name = giLEN\_PATIENT\_NAME*  
*'ID = giLEN\_ID*  
*'Drug = giLEN\_DRUG*  
*'TX Center = giLEN\_TX\_CENTER*  
*'Organ = giLEN\_ORGAN*

*'Create a 64 byte string from the various data elements to be saved  
'This string identifies the format of custom information. If the format  
'changes in a future version, then this ID can be used to determine which  
'version of the software saved the info to the device.*  
sCustomData = gsREV\_DATA\_STRUCTURE

*'Save the 2 digit number that represents this drug.  
'To save space, a numerical index of the Organ name is stored in the device*  
i = GetDrugRefNumber()  
sTemp = CStr(i)  
If Len(sTemp) < 2 Then sTemp = "0" + sTemp *'force code to be 2 digits*  
sCustomData = sCustomData + sTemp *'concatenate result to outbound string*

*'Save the 2 digit number that represents this organ  
'To save space, a numerical index of the Organ name is stored in the device*

## Comm.bas - Comm\_SendCustomData

36

```

i = GetOrganRefNumber()
sTemp = CStr(i)
If Len(sTemp) < 2 Then sTemp = "0" + sTemp    'force code to be 2 digits
sCustomData = sCustomData + sTemp

sTemp = DataStruct.sPatientID
If Len(sTemp) > giLEN_ID Then
    sTemp = Left$(sTemp, giLEN_ID)            'name is too long, trim letters of first name
ElseIf Len(sTemp) < giLEN_ID Then
    sTemp = sTemp + Space$(giLEN_ID - Len(sTemp))    'name is too short to fill the designated length
End If
sCustomData = sCustomData + sTemp                'concatenate result to outbound string

sTemp = DataStruct.sTxCenter
If Len(sTemp) > giLEN_TX_CENTER Then
    sTemp = Left$(sTemp, giLEN_TX_CENTER)        'name is too long, trim letters of first name
ElseIf Len(sTemp) < giLEN_TX_CENTER Then
    sTemp = sTemp + Space$(giLEN_TX_CENTER - Len(sTemp))    'name is too short to fill the designated length
End If
sCustomData = sCustomData + sTemp                'concatenate result to outbound string

sTemp = DataStruct.sPatientLastName + "." + DataStruct.sPatientFirstName
If Len(sTemp) > giLEN_PATIENT_NAME Then
    sTemp = Left$(sTemp, giLEN_PATIENT_NAME)    'name is too long, trim letters of first name
ElseIf Len(sTemp) < giLEN_PATIENT_NAME Then
    sTemp = sTemp + Space$(giLEN_PATIENT_NAME - Len(sTemp))    'name is too short to fill the designated length
End If
sCustomData = sCustomData + sTemp                'concatenate result to outbound string

```

'Assemble the string to be sent

Select Case sLocation

Case DATA\_BEGIN\_CUSTOM1

sOut = "Vw"

sData = Mid\$(sCustomData, 1, 16)

Case DATA\_BEGIN\_CUSTOM2

sOut = "Xx"

sData = Mid\$(sCustomData, 17, 16)

Case DATA\_BEGIN\_CUSTOM3

sOut = "Yy"

sData = Mid\$(sCustomData, 33, 16)

Case DATA\_BEGIN\_CUSTOM4

sOut = "Zz"

sData = Mid\$(sCustomData, 49, 16)

End Select

'Since the device string is limited to 16 chars, ensure that only the 1st 16 chars  
'of the string are sent.

sOut = sOut + sData

'add data string to command

CreateChecksum sOut, sChecksum

'calculate a checksum

sOut = sOut + sChecksum + "I"

'append checksum and ending string identifier

Comm\_SendDataToDevice (sOut)

'send string to comm port

r = Comm\_GetDeviceReply(sReply, IErrorCode)

If sReply = "\$" Then

'string was successfully interpreted by device

Comm\_SendCustomData = True

'return success to caller

ElseIf sReply = "?" Then

'string was not interpreted properly

IReturnError = ERR\_COMM\_BADRESPONSE

Else

IReturnError = IErrorCode

End If

End Function

## Comm.bas - Comm\_SendCustomData

37

**Public Function Comm\_SendDosingParams(DataStruct As DeviceDataStruct, IReturnError As Long)**  
*Sends the dosing parameters from the global structure to the device*

*'Structure: "Ddtuuvvwwxyzzmmss"*  
*'Response: "\$"*  
*'"tt" is Dose Size in pump ticks. Note that pump ticks per milliliter is fixed at 40. Hex value from 0 to 0xff. (Maximum dose size is currently 5 ml or 200 decimal)*  
*'"uu", "vv", "ww", "xx" are four Dose Interval values in hours between doses. Hex values between 1 and 0xff.*  
*'"yy" is Number of Doses per day. Hex value from 1 to 4.*  
*'"zz" is Pump Ticks per 10 mg Conversion value. Hex value. Typical value with present medication is 4 ticks per 10 milligrams.*  
*'"mm" is Lockout Hours value. Number of hours to prevent dosing after a dose is taken.*  
*'"ss" are the two checksum digits (hex) equal the one's complement value of the two's complement sum of the command characters and the data. The ASCII values are simply added together in an 8 bit sum, then one is subtracted (modulo 255).*

Dim iData As Integer, sData As String, sOut As String, sReply As String  
 Dim r As Integer, IErrorCode As Long, sChecksum As String, i As Integer  
 Dim iLastIntervalSet As Integer

On Error GoTo 0

sOut = "Dd" *'put command in string*

*'Get Dose Size in pump ticks*  
 iData = Val(DataStruct.sDoseSize) \* 40 / 100 *'get dose size from global struct & convert to pump ticks (convert from mg to ml)*  
 sData = CStr(Hex(iData)) *'convert value to a hex string*  
 If Len(sData) < 2 Then sData = "0" + sData *'ensure string is always 2 bytes long*  
 sOut = sOut + sData

iLastIntervalSet = 1 *'1:00 am is the ref time for the first dose*

*'Get Dose intervals in hours*  
 For i = 1 To giMaxDoseTimes *'max doses per day*  
   iData = 0 *'in case of conversion error, reset temp value*  
   If DataStruct.dPrescribedDoseTime(i) > 0 Then *'a negative number indicates no time was set*  
     iData = Format\$(DataStruct.dPrescribedDoseTime(i), "hh") *'convert fractional day to hours*  
     iData = iData - iLastIntervalSet *'this time is relative tot he last interval that was set*  
     *'change time to midnight*  
     If iData < 0 Then iData = 24 - Abs(iData)  
     iLastIntervalSet = Format\$(DataStruct.dPrescribedDoseTime(i), "hh") *'the next interval is relative to the last one that is set*  
   End If  
   sData = CStr(Hex(iData)) *'convert value to a hex string*  
   If Len(sData) < 2 Then sData = "0" + sData *'ensure string is always 2 bytes long*  
   sOut = sOut + sData  
 Next i

*'Get number of doses per day*  
 sData = CStr(Hex(DataStruct.iDosesPerDay)) *'convert value to a hex string*  
 If Len(sData) < 2 Then sData = "0" + sData *'ensure string is always 2 bytes long*  
 sOut = sOut + sData

*'Get conversion value*  
 iData = 0 *'in case of error, reset temp value*  
 iData = CInt(DataStruct.sDoseResolution) *'get dose resolution from global struct*  
 sData = CStr(Hex(iData)) *'convert value to a hex string*  
 If Len(sData) < 2 Then sData = "0" + sData *'ensure string is always 2 bytes long*  
 sOut = sOut + sData

*'Get Dose lockout hours*  
 iData = 0 *'in case of error, reset temp value*  
 iData = CInt(DataStruct.sDoseLockoutHours) *'get lockout hour from global struct*  
 sData = CStr(Hex(iData)) *'convert value to a hex string*  
 If Len(sData) < 2 Then sData = "0" + sData *'ensure string is always 2 bytes long*  
 sOut = sOut + sData

CreateChecksum sOut, sChecksum *'calculate a checksum*  
 sOut = sOut + sChecksum + "!" *'append checksum and ending string identifier*

## Comm.bas - Comm\_SendDosingParams

38

```

Comm_SendDataToDevice (sOut)           'send string to comm port
r = Comm_GetDeviceReply(sReply, IErrorCode)
If sReply = "$" Then                   'string was successfully interpreted by device
    Comm_SendDosingParams = True       'return success to caller
ElseIf sReply = "?" Then               'string was not interpreted properly
    IReturnError = ERR_COMM_BADRESPONSE
Else
    IReturnError = IErrorCode
End If

```

End Function

**Public Function Comm\_SendSerialNumber(DataStruct As DeviceDataStruct, IReturnError As Long) As Integer**  
*'Sends the serial number from the global structure to the device*

```

Dim sData As String, sOut As String, sReply As String, sChecksum As String
Dim r As Integer, IErrorCode As Long

sData = Left$(Trim(DataStruct.sSerialNumber), 10) 'trim leading spaces and use first 16 chars
sData = sData + Space$(10 - Len(sData))          'pad the string with spaces
sOut = "Nn" + sData                               'add data string to command
CreateChecksum sOut, sChecksum                    'calculate a checksum
sOut = sOut + sChecksum + "I"                     'append checksum and ending string identifier
Comm_SendDataToDevice (sOut)                     'send string to comm port
r = Comm_GetDeviceReply(sReply, IErrorCode)
If sReply = "$" Then                               'string was successfully interpreted by device
    Comm_SendSerialNumber = True                   'return success to caller
ElseIf sReply = "?" Then                           'string was not interpreted properly
    IReturnError = ERR_COMM_BADRESPONSE
Else
    IReturnError = IErrorCode
End If

```

End Function

**Private Sub ConvertHexStringToAscii(ByVal sData, sConverted As String)**  
*'String Data from the device is usually returned as Hex characters.*  
*'Convert the string (passed in here) to an ASCII string and return to caller.*  
*'Such strings are items like patient name, serial number, etc.*

```

Dim sTemp As String, i As Integer, iTemp As Integer
On Error Resume Next
sConverted = "" 'clear out any old string
For i = 1 To Len(sData) Step 2
    sTemp = "&H" + Mid$(sData, i, 2) 'get a 2 char hex byte from string
    sTemp = Chr$(sTemp)               'convert value to ASCII
    sConverted = sConverted + sTemp   'concatenate to existing string being built
Next i

```

End Sub

## Comm.bas - CreateChecksum

39

**Private Sub CreateChecksum(sOut As String, sChecksum As String)***'The string "sOut" will be sent to the device by another procedure. Before it is sent.**'this procedure calculates a checksum and returns it to the caller**'Return the ASCII representation of the checksum value.*

Dim i As Integer, iChecksumTally As Long, iChecksumByteLow As Integer

For i = 1 To Len(sOut)

*'calculate checksum*

iChecksumTally = iChecksumTally + Asc(Mid\$(sOut, i, 1))

Next i

iChecksumByteLow = iChecksumTally Mod 256

iChecksumByteHigh = iChecksumTally \ 256 *'not being using by the device*sChecksum = Hex(iChecksumByteLow - 1) *'checksum is the "one's complement value of a two's complement checksum"**'value must always be 2 chars*If Len(sChecksum) < 2 Then sChecksum = "0" + sChecksum *'place a leading "0" in front of checksum*

End Sub

**Public Sub EstablishDeviceComm()***'This procedure continues to try and establish communication with the Device**'until it succeeds. When successful, control is returned to the calling procedure.**'The purpose of this procedure is to allow the user to try cable changes, device movement, etc. without having to continue pressing keys on the keyboard.*

Dim r As Integer, iErrorCode As Long

QueryDevice:

r = Comm\_CheckComm(iErrorCode)

If r &lt;&gt; True Then

DoEvents *'allow other Windows events to be processed, so we don't lock up the computer*Wait 1 *'wait an additional amount of time before trying*GoTo QueryDevice *'try comm again*

End If

End Sub

**Function Comm\_InitializeCommPort() As Integer***'Get the initial values from INI file and**'Initialize device comm port settings*

Dim iReply As Long

Const sSection = "Communications"

*'Get the amount of time needed for a reply to be received from the device*

giDeviceResponseWait = CInt(GetINISetting(gsAppIniFileSpec, sSection, "Device Response Wait", "50"))

If giDeviceResponseWait < 25 Then giDeviceResponseWait = 25 *'set a minimum delay time*If giDeviceResponseWait > 500 Then giDeviceResponseWait = 500 *'clamp upper limit*frmMain.CommTimer.Interval = giDeviceResponseWait *'set up the timer**'Get the comm port speed settings*

giCommPort = CInt(GetINISetting(gsAppIniFileSpec, sSection, "Port", "1"))

If giCommPort = 0 Then giCommPort = 2 *'set a default of comm 2 if nothing is in the file*

gsCommDeviceSettings = GetINISetting(gsAppIniFileSpec, sSection, "Settings", "2400,n,8,2")

*'prevent device unavailable error*

If frmMain.CommDevice.PortOpen = True Then frmMain.CommDevice.PortOpen = False

*'close port if open*

frmMain.CommDevice.Settings = gsCommDeviceSettings

frmMain.CommDevice.CommPort = CStr(giCommPort)

## Comm.bas - Comm\_InitializeCommPort

40

```
frmMain.CommDevice.InBufferSize = 1024
frmMain.CommDevice.InputLen = 0
Comm_InitializeCommPort = True           'return success to caller
```

```
Comm_InitializeCommPort_Exit:
Exit Function
```

```
Comm_InitializeCommPort_Error:
Comm_InitializeCommPort = Err           'return error to caller
On Error GoTo 0
Resume Comm_InitializeCommPort_Exit
```

```
End Function
```

## Sub Device\_OnComm()

*This procedure is called by the OnComm event of the comm control located on frmMain. This is so that the code can be shared between applications.*

```
Dim r As Integer, sTemp As String
```

```
r = frmMain.CommDevice.CommEvent
```

```

If r = MSCOMM_ER_RXOVER Then
    'An over run error occurred. Usually happens when getting events.
    Exit Sub
End If

If r = MSCOMM_EV_EOF Then 'end of file flag received
    Exit Sub              'who cares. Happens during receipt of events
End If

If r = MSCOMM_ER_BREAK Then 'break signal received
    Exit Sub
End If

If r = 3 Or r = 4 Or r = 5 Then 'rts, xon xoff, CD error
    Exit Sub
End If

MsgBox "Unexpected error occurred with the device. Please try again.", , "Comm Event - " + Str$(r)
```

Comm.bas - Device\_OnComm

41

End Sub

**Private Sub InterpretScoreData(DataStruct As DeviceDataStruct, ByVal sData As String, ByVal iCurrentDayPoi**  
*'Get score data and place into global structure. Seems that the pointer is 0 based.*  
*'The value is from 0 to 4 doses taken per day. A value of "&HFF" means that the value*  
*'is cleared and not set yet.*

Dim i As Integer, sTemp As String, iTempPointer As Integer

iTempPointer = iCurrentDayPointer + 1 *'start with the current day*

For i = 1 To 14 *'get 14 days of data*  
 sTemp = Mid\$(sData, iTempPointer + 1, 2) *'get score for this day (2 byte hex)*  
 DataStruct.sScoreData(i) = sTemp *'save score*  
 iTempPointer = iTempPointer + 2  
 If iTempPointer > 13 Then iTempPointer = 1 *'end of circular buffer. Start at bottom*  
Next i

End Sub

**Private Function ParseMemoryContents(DataStruct As DeviceDataStruct, sAllData As String, IErrorCode As Lo**  
*On Error Resume Next*

Dim i As Integer, sDoseData As String, iChecksum As Long  
 Dim sTemp As String, sConverted As String, iTemp As Long, r As Integer, sCustomData As String  
 Dim sLastIntervalTime As String, iStartingLocation As Integer  
 ReDim sTempList(25) As String

On Error GoTo ParseMemoryContents\_Error

*'Get Clock*

*'The device clock reports the number of 10 minute intervals that have passed since 1 am on*  
*'the morning of the first dose. Thus, calculate when the first dose occurred for later use.*  
 sTemp = "&H" + "0" + Mid\$(sAllData, DATA\_BEGIN\_CLOCK + 2, 2) + Mid\$(sAllData, DATA\_BEGIN\_CLOCK, 2)  
*'calculate back to the date and time the clock should have started.*  
 DataStruct.dDeviceRefDateTime = CDb1(DateAdd("n", -(CLng(sTemp)) \* 10, Now))  
 DataStruct.iDeviceInitDate = CLng(DataStruct.dDeviceRefDateTime)

*'Get Battery Change Timer*

sTemp = "&H" + "0" + Mid\$(sAllData, DATA\_BEGIN\_BATTERY\_CHANGE\_TIMER + 2, 2) + Mid\$(sAllData,  
 DATA\_BEGIN\_BATTERY\_CHANGE\_TIMER, 2)  
 iTemp = CLng(sTemp) / 6 *'convert to hours (there are 6 "ten-minute" intervals in one hour)*  
 DataStruct.sBatteryChangeTimer = Format\$(CStr(iTemp / 24), "#0.0 days") *'convert to days and store it*

*'Get Dosing Event Data*

sTemp = "&H" + "0" + Mid\$(sAllData, DATA\_BEGIN\_COMPLIANCE\_POINTER + 2, 2) + Mid\$(sAllData,  
 DATA\_BEGIN\_COMPLIANCE\_POINTER, 2) *'get pointer*  
 iTemp = CLng(sTemp) *'convert hex value to a long value*  
 If iTemp Then sDoseData = Mid\$(sAllData, DATA\_BEGIN\_COMPLIANCE\_DATA, (iTemp - 132) \* 2) *'get string*

*'Get Compliance Checksum*

sTemp = "&H" + "0" + Mid\$(sAllData, DATA\_BEGIN\_COMPLIANCE\_CHECKSUM + 2, 2) + Mid\$(sAllData,  
 DATA\_BEGIN\_COMPLIANCE\_CHECKSUM, 2)  
 iChecksum = CLng(sTemp)  
 r = InterpretDosingData(DataStruct, sDoseData, iTemp, iChecksum, IErrorCode) *'parse out the events and place in global structure*  
 If r = False Then *'bad checksum data*  
 GoTo ParseMemoryContents\_Exit  
End If

*'Get Patient Score Data*

sTemp = Mid\$(sAllData, DATA\_BEGIN\_SCORE, 28)  
*'parse the 14 days of dosing and store in global structure*  
 If Len(sTemp) > 0 Then  
 iTemp = CLng("&H0" + Mid\$(sAllData, DATA\_BEGIN\_DOSE\_SCORE\_DAY\_POINTER, 2))



## Comm.bas - ParseMemoryContents

42

```
If ITemp Then InterpretScoreData DataStruct, sTemp, CInt(ITemp) 'parse out the scores and place in global structure
End If
```

```
'Get Error Flags
```

```
sTemp = Mid$(sAllData, DATA_BEGIN_ERROR_FLAGS, 2)
```

```
If Len(sTemp) > 0 Then InterpretErrorFlags DataStruct, Val(sTemp) 'parse out the flags and save in global structure
```

```
'Get Medication remaining in device
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_MED_REMAINING + 2, 2) + Mid$(sAllData, DATA_BEGIN_MED_REMAINING, 2)
DataStruct.sMedRemaining = CStr(CSng(sTemp / 40) * 100) + " mg" 'pump ticks are fixed at 40 per milliliter (100 mg per ml)
```

```
'Get Dose Lockout Hours
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_LOCKOUT_HOURS, 2)
DataStruct.sDoseLockoutHours = CStr(CSng(sTemp))
```

```
'Get Doses per day
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSES_PER_DAY, 2)
DataStruct.iDosesPerDay = CInt(sTemp)
```

```
'Get Dose Resolution
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_CONVERSION, 2)
DataStruct.sDoseResolution = CStr(CSng(sTemp))
```

```
'Get Dose Intervals
```

```
sLastIntervalTime = "1:00"
```

```
'Get Dose Interval 1 (alarm time)
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_INTERVAL1, 2)
```

```
If Val(sTemp) Then
```

```
sTemp = DateAdd("H", CDBl(sTemp), sLastIntervalTime) 'the first dose is relative to 1:00 am
```

```
sLastIntervalTime = sTemp
```

```
DataStruct.dPrescribedDoseTime(1) = TimeValue(sTemp)
```

```
Else
```

```
DataStruct.dPrescribedDoseTime(1) = -1 'this value indicates that no time was received
```

```
End If
```

```
'Get Dose Interval 2 (alarm time)
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_INTERVAL2, 2)
```

```
If Val(sTemp) Then
```

```
sTemp = DateAdd("H", CDBl(sTemp), sLastIntervalTime) 'the first dose is relative to 1:00 am
```

```
sLastIntervalTime = sTemp
```

```
DataStruct.dPrescribedDoseTime(2) = TimeValue(sTemp)
```

```
Else
```

```
DataStruct.dPrescribedDoseTime(2) = -1 'this value indicates that no time was received
```

```
End If
```

```
'Get Dose Interval 3 (alarm time)
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_INTERVAL3, 2)
```

```
If Val(sTemp) Then
```

```
sTemp = DateAdd("H", CDBl(sTemp), sLastIntervalTime) 'the first dose is relative to 1:00 am
```

```
sLastIntervalTime = sTemp
```

```
DataStruct.dPrescribedDoseTime(3) = TimeValue(sTemp)
```

```
Else
```

```
DataStruct.dPrescribedDoseTime(3) = -1 'this value indicates that no time was received
```

```
End If
```

```
'Get Dose Interval 4 (alarm time)
```

```
sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_INTERVAL4, 2)
```

```
If Val(sTemp) Then
```

```
sTemp = DateAdd("H", CDBl(sTemp), sLastIntervalTime) 'the first dose is relative to 1:00 am
```

```
sLastIntervalTime = sTemp
```

```
DataStruct.dPrescribedDoseTime(4) = TimeValue(sTemp)
```

```
Else
```

```
DataStruct.dPrescribedDoseTime(4) = -1 'this value indicates that no time was received
```

```
End If
```

```
'Get Dose Size
```

## Comm.bas - ParseMemoryContents

43

```

sTemp = "&H0" + Mid$(sAllData, DATA_BEGIN_DOSE_SIZE, 2)
If IsNumeric(sTemp) Then
    DataStruct.sDoseSize = CStr(CSng(sTemp) / 40 * 100)    'convert from mg to ml
Else
    DataStruct.sDoseSize = ""
End If

```

*'There are 4 fields in the device containing 16 characters each. In the original device design, this was intended to contain 4 separate pieces of information. The client has now decided that some fields are too short and others are too long. Thus, the fields are combined to be one string of 64 characters.*

```

'Patient name = giLEN_PATIENT_NAME
'ID           = giLEN_ID
'Drug         = giLEN_DRUG
'TX Center    = giLEN_TX_CENTER
'Organ        = giLEN_ORGAN

```

*'Send a message to the user if the revision level is higher than the one this software is using to send custom data to the device. The user must upgrade to the current version in order to get accurate custom data. This code should also handle any previous versions that saved data to the device.*

```

'Get Custom string 1
sTemp = Mid$(sAllData, DATA_BEGIN_CUSTOM1, 32)
ConvertHexStringToAscii sTemp, sConverted
sCustomData = sConverted

```

```

'Get Custom string 2
sTemp = Mid$(sAllData, DATA_BEGIN_CUSTOM2, 32)
ConvertHexStringToAscii sTemp, sConverted
sCustomData = sCustomData + sConverted

```

```

'Get Custom string 3
sTemp = Mid$(sAllData, DATA_BEGIN_CUSTOM3, 32)
ConvertHexStringToAscii sTemp, sConverted
sCustomData = sCustomData + sConverted

```

```

'Get Custom string 4
sTemp = Mid$(sAllData, DATA_BEGIN_CUSTOM4, 32)
ConvertHexStringToAscii sTemp, sConverted
sCustomData = sCustomData + sConverted

```

```

'Pull apart the 64 char string into its sub-components
'Get the custom data structure revision level that was previously saved to the device.
'Note: this is not the same as the major and minor versions of the host software.
iStartingLocation = 1
sTemp = Mid$(sAllData, iStartingLocation, giLEN_REV_DATA_STRUCTURE)
ConvertHexStringToAscii sTemp, sConverted
'The device custom data was apparently saved with a newer version of software than this one.
If Val(sConverted) > gsREV_DATA_STRUCTURE Then
    IErrorCode = ERR_NEWER_HOST_SOFTWARE
    GoTo ParseMemoryContents_Exit
End If

```

```

'Determine the real name of the Drug by the reference number received from the device
iStartingLocation = iStartingLocation + giLEN_REV_DATA_STRUCTURE
sTemp = Trim(Mid$(sCustomData, iStartingLocation, giLEN_DRUG))
r = Val(sTemp)
If r > 0 And r < UBound(gsDrugNames) Then DataStruct.sDrug = gsDrugNames(r)

```

```

'Determine the real name of the Organ by the reference number received from the device
iStartingLocation = iStartingLocation + giLEN_DRUG
sTemp = Trim(Mid$(sCustomData, iStartingLocation, giLEN_ORGAN))
r = Val(sTemp)
If r > 0 And r < UBound(gsOrganNames) Then DataStruct.sOrgan = gsOrganNames(r)

iStartingLocation = iStartingLocation + giLEN_ORGAN

```

## Comm.bas - ParseMemoryContents

44

```

DataStruct.sPatientID = Trim(Mid$(sCustomData, iStartingLocation, giLEN_ID))

iStartingLocation = iStartingLocation + giLEN_ID
DataStruct.sTxCenter = Trim(Mid$(sCustomData, iStartingLocation, giLEN_TX_CENTER))

iStartingLocation = iStartingLocation + giLEN_TX_CENTER
r = ParseDelimString(Trim(Mid$(sCustomData, iStartingLocation, giLEN_PATIENT_NAME)), ",", sTempList())
DataStruct.sPatientLastName = Trim$(sTempList(1))
DataStruct.sPatientFirstName = Trim$(sTempList(2))

'Get Serial Number
sTemp = Mid$(sAllData, DATA_BEGIN_SERIAL_NUMBER, 20)
ConvertHexStringToAscii sTemp, sConverted
DataStruct.sSerialNumber = Trim(sConverted)
ParseMemoryContents = True 'send success to caller

ParseMemoryContents_Exit:
On Error GoTo 0
Exit Function

ParseMemoryContents_Error:
'force a checksum error here because any type of error is likely due to a checksum problem
iErrorCode = ERR_COMM_CHECKSUM
Resume ParseMemoryContents_Exit

End Function

```

## Public Sub PollDeviceContinually(SourceForm As Form)

*'This procedure continues to try and establish communication with the Device  
'until it succeeds. When successful, control is returned to the calling procedure.  
'The purpose of this procedure is to allow the user to try cable changes, device  
'movement, etc. without having to continue pressing keys on the keyboard.*

Dim r As Integer, bProcedureInProgress As Boolean, iErrorCode As Long

```

If bProcedureInProgress Then Exit Sub
If gbCommBusy Or gbCommReplyPending Then Exit Sub
bProcedureInProgress = True 'prevent recursive calls to this procedure

```

## QueryDevice:

```

DoEvents 'allow other Windows events to be processed, so we don't lock up the computer
If gbCommOK = True Then 'no need to poll as often if device was working the last time we checked
    Wait 5 'wait an additional amount of time before trying
Else
    Wait 0.05 'poll faster until a good comm is made
End If

```

```

If Not gbCommBusy And Not gbCommReplyPending Then 'poll only if port not busy

```

```

    If gbKeepPollingDevice = False Then Exit Sub
    SourceForm.imgPolling.Visible = True
    SourceForm.imgPolling.Refresh
    r = Comm_CheckComm(iErrorCode)
    If gbKeepPollingDevice = False Then Exit Sub
    If r = True Then 'comm is working
        'display status has not yet been updated
        If Not gbCommOK Then DisplayCommOk SourceForm

```

```

    Else 'comm is NOT working
        'display status has not yet been updated
        If gbCommOK Then DisplayCommError SourceForm

```

```

    End If
End If

```

```

Wait 0.05 'allow polling icon to be viewed
If gbKeepPollingDevice = False Then Exit Sub

```

## Comm.bas - PollDeviceContinually

45

SourceForm.imgPolling.Visible = False

If r = ERR\_COMM\_TIMEOUT Then 'send an additional error message that no reply was received  
MsgBox "No response was received to a wake up command that was sent to the DosPro device. ", , "Communication Error"  
End If

'flag has not been reset yet  
If gbKeepPollingDevice Then GoTo QueryDevice 'try comm again

bProcedureInProgress = False 'allow future calls to this procedure now that we are finished

End Sub

### Public Sub Comm\_SendDataToDevice(ByVal sOut As String)

Dim dGoAheadTime As Double  
'A data string should have been assembled by another procedure and is now  
'ready to sent to the device.

'if another command is in progress then wait till it is done  
'if the flags have not been reset after this delay, then exit loop anyway  
'This prevents lockups inside this loop in case there is a problem elsewhere  
dGoAheadTime = DateAdd("s", 5, CDBl(Now))

Do While gbCommBusy Or gbCommReplyPending  
DoEvents  
DoEvents  
DoEvents  
If CDBl(Now) > dGoAheadTime Then Exit Do  
Loop

gbCommBusy = True 'set busy flag (gets reset if timeout or reply not received)  
'if comm port is not open then open it  
If Not frmMain.CommDevice.PortOpen Then frmMain.CommDevice.PortOpen = True  
frmMain.CommDevice.InputLen = 0 'clear input buffer  
frmMain.CommDevice.Output = sOut 'send string to device

End Sub

### Private Sub SetCommTimer(iTime As Integer)

'The comm timer determines whether or not a reply has come back from the device.  
'The timer fires an event if the iTime has passed without the timer being reset.  
'Reset the timer to the interval passed in, then start it.  
'Set the comm busy flag, then return to caller

frmMain.CommTimer.Enabled = False 'disable timer while resetting it  
frmMain.CommTimer.Interval = iTime 'set interval  
gbCommTimerExpired = False 'reset timer expiration flag  
frmMain.CommTimer.Enabled = True 'start timer

End Sub

## Comm.bas - SetCommTimer

46

Attribute VB\_Name = "modPrinting"  
Option Explicit

Public gbPrintFormLoading As Integer  
Public gbPrinterErrorDetected As Integer  
Public giTotalPrintPages As Integer 'track the number of pages being previewed  
Public gbPreventPreviewUpdates As Integer  
Public giPrintedPageNumber As Integer  
Public gbPageNumberSuspend As Integer 'if true, don't print page number for the active page (probably cover pictures)  
  
Public giFontOptSel As Integer  
Public gsSelectPrintPages As String  
Public gbPrinterErrorReceived As Integer 'tells other procs that error occurred Proc must reset flag  
Public gbPrintSpoolingInProgress As Integer 'prevent crashes during spooling

## Private Sub btnPrinter\_Preview\_Click\_Proc()

```

    If giTotalPrintPages > 1 Then 'there is more than one page to print
        frmSelectPages.Show MODAL
        Select Case gsSelectPrintPages
            Case "All"
                frmPrint.MousePointer = vbHourglass
                gbPrintSpoolingInProgress = True
                frmPrint.vsPrinter1.Action = paPrintAll 'print all pages
                gbPrintSpoolingInProgress = False

            Case "Page"
                frmPrint.MousePointer = vbHourglass
                gbPrintSpoolingInProgress = True
                frmPrint.vsPrinter1.Action = paPrintPage 'print current page only
                gbPrintSpoolingInProgress = False

            Case ""
                'nothing to do
                gbPrintSpoolingInProgress = False
        End Select

    Else 'printing a single page that is not a picture
        frmPrint.MousePointer = vbHourglass
        gbPrintSpoolingInProgress = True
        frmPrint.vsPrinter1.Action = paPrintAll 'print all pages
        gbPrintSpoolingInProgress = False
    End If

```

Printing.bas - btnPrinter\_Preview\_Click .vbs

47

```

frmPrint.btnClose.Enabled = True
frmPrint.btnPrintNow.Enabled = True
frmPrint.MousePointer = vbDefault
DoEvents

```

*'allow button to show*

End Sub

**Private Sub DrawHorizontalLine(cPrinter As Control, lPenColor As Long)**

```

    frmPrint.vsPrinter1.FontSize = dgBodyTextSize
    frmPrint.vsPrinter1 = ""
    Exit Sub

```

*'skip a line from above*

*'Draw a horizontal divider on the page  
'Usually divides the header or topic from the rest of the page*

```

cPrinter.FontSize = 12
cPrinter = ""
cPrinter.PenStyle = 0
cPrinter.PenWidth = 10
cPrinter.PenColor = lPenColor
cPrinter.BrushColor = lPenColor

```

*'skip a line from above*

*'0=solid 2=dot*

*'set pen width*

*'set pen color*

```

    'Print line only across a portion of page
    cPrinter.X1 = (cPrinter.PageWidth / 2) - (cPrinter.PageWidth * 0.25)
    cPrinter.X2 = (cPrinter.PageWidth / 2) + (cPrinter.PageWidth * 0.25)

```

```

    cPrinter.Y1 = cPrinter.CurrentY
    cPrinter.Y2 = cPrinter.CurrentY + 50

```

```

    cPrinter.Draw = 2

```

*'1=line, 2=rectangle*

End Sub

**Private Sub PrintAllPatientsSummary()**

```

    Dim l As Integer, lErrorCode As Long, sTableFormat As String, sTable As String, sList As String
    Dim fFontSize As Single, lCount As Integer

```

On Error Resume Next

*'Prepare progress gauge*

With frmPrint

.pnlProgress.FloodPercent = 0

.pnlProgressContainer.Visible = True

.pnlProgressContainer.Refresh

End With

InitPageProperties

fFontSize = 10

*'Print the logo on the first page*

With frmPrint.vsPrinter1

.X1 = 700

.X2 = frmPrint.vsPrinter1.X1 + frmPrint.picLogo.Width

.Y1 = 500

.Y2 = frmPrint.vsPrinter1.Y1 + frmPrint.picLogo.Height

.Picture = frmPrint.picLogo.Picture

End With

*'Print Information Header*

gbPageNumberSuspend = False

With frmPrint.vsPrinter1

64

## Printing.bas - PrintAllPatientsSummary

48

```

.FontName = "Arial"
.FontBold = True
.TextAlign = taCenterTop 'center text used in paragraphs
.CurrentY = 1440 * 1 'print name on this line
.FontSize = fFontSize * 1.6 'set font size
.FontItalic = True
frmPrint.vsPrinter1 = "All Patient's Summary" 'print name
.FontSize = fFontSize * 1.2 'set font size
.FontItalic = False
DrawHorizontalLine frmPrint.vsPrinter1, &H40000 'Draw a color line
frmPrint.vsPrinter1 = "" 'skip a line
frmPrint.vsPrinter1 = frmAllPatients.cmbDataToView.Text + " with " + frmAllPatients.Label1(0).Caption
frmPrint.vsPrinter1 = "Date Range: " + frmAllPatients.cmbDateSelection.Text + " from " + frmAllPatients.txtStartDate.Value + " to "
+ frmAllPatients.txtEndDate.Value

frmPrint.vsPrinter1 = "" 'skip a line
.TextAlign = taCenterTop
.FontBold = False
.TableBorder = tbNone
.FontSize = fFontSize * 1.1 'set font size
.Table = sTable 'send out table

frmPrint.vsPrinter1 = "" 'skip a line
frmPrint.vsPrinter1 = "" 'skip a line
.LineSpacing = 90 '% of current font
.TextAlign = taCenterTop 'center text
End With

'Print the report Data
sTableFormat = "<2400|<1800|>1700|>1700|>1000;"
'Get the column titles from grid
With frmAllPatients.grid
.Row = 0
.Col = 0
sList = .Text
.Col = 1
sList = sList + "|" + .Text
.Col = 2
sList = sList + "|" + .Text
.Col = 3
sList = sList + "|" + .Text
.Col = 4
sList = sList + "|" + .Text
End With
sTable = sTableFormat + sList

With frmPrint.vsPrinter1
.TableBorder = tbBottom
.FontSize = 10
.FontBold = True
.Table = sTable 'send out header
frmPrint.vsPrinter1 = ""
End With

'Print the information from the grid control
With frmAllPatients.grid
iCount = .Rows - 1
sList = ""
For i = 1 To iCount 'number of patients in grid
.Row = i
.Col = 0
sList = sList + .Text
.Col = 1
sList = sList + "|" + .Text
.Col = 2
sList = sList + "|" + .Text
.Col = 3
sList = sList + "|" + .Text

```

## Printing.bas - PrintAllPatientsSumm

49

```

.Col = 4
sList = sList + "]" + .Text + ":-"

frmPrint.pnlProgress.FloodPercent = (i / iCount) * 100
frmPrint.pnlProgressContainer.Refresh
Next i
End With

sTable = sTableFormat + sList
With frmPrint.vsPrinter1
.FontSize = fFontSize * 0.9 'set font size
.LineSpacing = 80 ' % of current font
.TextAlign = taCenterTop
.TableBorder = tbNone
.Table = sTable 'send out table
End With

On Error GoTo 0
frmPrint.pnlProgressContainer.Visible = False 'turn off progress indicator
frmPrint.pnlProgressContainer.Refresh

End Sub

```

## Private Sub PrintPatientDosingReport()

```

Dim i As Integer, IErrorCode As Long, sTableFormat As String, sTable As String, sList As String
Dim fFontSize As Single, iCount As Integer, bItemChecked As Boolean

On Error Resume Next

'Print Cover Art to the Print preview control if needed and available
If frmPrint.lbPictures.ListIndex > 0 Then 'a cover is chosen
If FileExists("covers\" + sgCurrentCoverName, IErrorCode) Then 'look for a bitmap on disk
'Print preview the Picture
gbPageNumberSuspend = True
LoadPicture ToPrinterControl True 'get cover
InitPageProperties
frmPrint.vsPrinter1.Action = 4 'start a new page
gbPageNumberSuspend = False
End If
End If

'Prepare progress gauge
With frmPrint
.pnlProgress.FloodPercent = 0
.pnlProgressContainer.Visible = True
.pnlProgressContainer.Refresh
End With
InitPageProperties
fFontSize = 10

'Print the logo on the first page
With frmPrint.vsPrinter1
.X1 = 700
.X2 = frmPrint.vsPrinter1.X1 + frmPrint.picLogo.Width
.Y1 = 500
.Y2 = frmPrint.vsPrinter1.Y1 + frmPrint.picLogo.Height
.Picture = frmPrint.picLogo.Picture
End With

'Print Information Header
gbPageNumberSuspend = False
With frmPrint.vsPrinter1
.FontName = "Arial"

```



## Printing.bas - PrintPatientDosingRe

50

```

.FontSize = fFontSize * 1.6 'set font size
.FontBold = True
.FontItalic = True
.TextColor =
.TextAlign = taCenterTop 'center text used in paragraphs
.CurrentY = 1440 * 1 'pnt name on this line
frmPrint.vsPrinter1 = "Patient Dosing Report" 'pnt name
.FontSize = fFontSize * 1.2 'set font size
.FontItalic = False
frmPrint.vsPrinter1 = PAT_DATA.sPatientLastName + ", " + PAT_DATA.sPatientFirstName
DrawHorizontalLine frmPrint.vsPrinter1, &H40000 'Draw a color line
End With

```

```

sTableFormat = "<1400|<2800|<1400|<2800;"
sTable = sTableFormat + gsCustomLblPatientID + ":@" + PAT_DATA.sPatientID + ":@"
sTable = sTable + gsCustomLblTxCenter + ":@" + PAT_DATA.sTxCenter + ":@"
sTable = sTable + gsCustomLblDrug + ":@" + PAT_DATA.sDrug + ":@"
sTable = sTable + gsCustomLblOrgan + ":@" + PAT_DATA.sOrgan + ":@"
With frmPrint.vsPrinter1
    frmPrint.vsPrinter1 = "" 'skip a line
    .TextAlign = taCenterTop
    .FontBold = True
    .TableBorder = tbNone
    .FontSize = fFontSize * 1.1 'set font size
    .Table = sTable 'send out table

    frmPrint.vsPrinter1 = "" 'skip a line
    frmPrint.vsPrinter1 = "" 'skip a line
    .LineSpacing = 90 '% of current font
    .TextAlign = taCenterTop 'center text

```

```

sList = "Events Types Shown: "
If frmPatientDosingReport.chkDoses.Value Then
    sList = sList + "Doses Taken"
    bitemChecked = True
End If

```

```

If frmPatientDosingReport.chkDoseChanged Then
    If bitemChecked Then sList = sList + " and "
    sList = sList + "Dose Size Changes"
    bitemChecked = True

```

```

If frmPatientDosingReport.chkUserDefined Then
    If bitemChecked Then sList = sList + " and "
    sList = sList + "User Entries"
    bitemChecked = True
End If

```

```

If Not bitemChecked Then
    sList = sList + "None"
End If

```

```

frmPrint.vsPrinter1 = "" 'skip a line
.TextAlign = taCenterTop
.FontSize = fFontSize * 0.9 'set font size
frmPrint.vsPrinter1 = sList
End If
End With

```

```

'Pnt the report Data
frmPrint.vsPrinter1 = "" 'skip a line
frmPrint.vsPrinter1.TextAlign = taCenterTop
PrintDosingEventsHeader sTableFormat
'Pnt the information from the grid control
With frmPatientDosingReport.grid
    iCount = .Rows - 1
    sList = ""
    For i = 1 To iCount 'number of patients in gnd

```

Printing.bas - PrintPatientDosingRe

51

```

        .Row = i
        .Col = 0
        sList = sList + .Text
        .Col = 1
        sList = sList + "|" + .Text
        .Col = 2
        sList = sList + "|" + .Text
        .Col = 3
        sList = sList + "|" + .Text
        .Col = 4
        sList = sList + "|" + .Text
        .Col = 5
        sList = sList + "|" + .Text + ";"
        frmPrint.pnlProgress.FloodPercent = (i / iCount) * 100
        frmPrint.pnlProgressContainer.Refresh
    Next i
End With

sTable = sTableFormat + sList
With frmPrint.vsPrinter1
    .FontSize = fFontSize * 0.9 'set font size
    .LineSpacing = 80           '% of current font
    .TextAlign = taCenterTop
    .Table = sTable            'send out table
End With

On Error GoTo 0
frmPrint.pnlProgressContainer.Visible = False 'turn off progress indicator
frmPrint.pnlProgressContainer.Refresh

End Sub

```

Sub PrintDosingEventsHeader(sTableFormat As String)

Dim sTable As String  
Dim fPrevFont As Single, bPrevBold As Boolean, sList As String

```

fPrevFont = frmPrint.vsPrinter1.FontSize

'Print the information from the grid control
'Pass table format back to caller
sTableFormat = "<2100|<1600|<1000|<1700|<1700|<1700;"
With frmPrint.pnlDosingReport.grid
    sList = ""
    .Row = 0
    .Col = 0
    sList = sList + .Text
    .Col = 1
    sList = sList + "|" + .Text
    .Col = 2
    sList = sList + "|" + .Text
    .Col = 3
    sList = sList + "|" + .Text
    .Col = 4
    sList = sList + "|" + .Text
    .Col = 5
    sList = sList + "|" + .Text + ";"
End With

```

```

sTable = sTableFormat + sList
frmPrint.vsPrinter1.TableBorder = tbBottom
frmPrint.vsPrinter1.FontSize = 10
frmPrint.vsPrinter1.FontBold = True
frmPrint.vsPrinter1.Table = sTable 'send out header
frmPrint.vsPrinter1 = ""

```

## Printing.bas - PrintDosingEventsHeader

52

```

'Put setting back to previous ones
frmPrint.vsPrinter1.TableBorder = tbNone
frmPrint.vsPrinter1.FontSize = fPrevFont
frmPrint.vsPrinter1.FontBold = bPrevBold

```

End Sub

## Public Sub LoadPictureToPrinterControl(ByVal bCover)

```

'Set the printer control to size a picture and copy
'picture from holding area to the print preview control.
'if the picture to be displayed is a cover.
'then the bCover flag should be set to true by caller.
'otherwise it is assumed to be a border.

Dim iPaperWidth%, iPaperHeight%, iNonPrintWidth%, iNonPrintHeight%
frmPrint.vsPrinter1.PhysicalPage = True      'set physical page to paper dimension
iPaperWidth = frmPrint.vsPrinter1.PageWidth  'determine size of paper
iPaperHeight = frmPrint.vsPrinter1.PageHeight
frmPrint.vsPrinter1.PhysicalPage = False     'return printer to printable area
iNonPrintWidth = (iPaperWidth - frmPrint.vsPrinter1.PageWidth) / 2
iNonPrintHeight = (iPaperHeight - frmPrint.vsPrinter1.PageHeight) / 2

If iNonPrintWidth < 350 Then iNonPrintWidth = 350 'make a minimum margin
If iNonPrintHeight < 350 Then iNonPrintHeight = 350 'make a minimum margin

```

```

frmPrint.vsPrinter1.X1 = iNonPrintWidth
frmPrint.vsPrinter1.X2 = frmPrint.vsPrinter1.PageWidth - iNonPrintWidth
frmPrint.vsPrinter1.Y1 = iNonPrintHeight
frmPrint.vsPrinter1.Y2 = frmPrint.vsPrinter1.PageHeight - iNonPrintHeight
' frmPrint.vsPrinter1.Draw = 2 'picture holder only

```

```

frmPrint.vsPrinter1.Picture = LoadPicture("graphics\1" & "deco.wmf")

```

End Sub

## Private Sub InitPageMargins()

```

'Set margins
'Margins don't seem to set properly until the next page is created.
'That's why they can be set only once before printing begins.

frmPrint.vsPrinter1.MarginTop = 1350      'top margin
frmPrint.vsPrinter1.MarginBottom = 1500   'bottom margin

frmPrint.vsPrinter1.MarginLeft = 1725     'left margin
frmPrint.vsPrinter1.MarginRight = 1700    'right margin (from right edge)

```

End Sub

## Printing.bas - InitPageProperties

53

## Private Sub InitPageProperties()

```

'Reset margins for text and initialize other items
frmPrint.vsPrinter1.LineSpacing = 100      '100% of current font
'Set the normal attributes here
.
.
frmPrint.vsPrinter1.TextAlign = 0          'set centering back to normal

```

End Sub

## Private Sub PrintPageDate()

```

'Print Date
Dim lTextHeight As Long, lTextWidth As Long, sText$
'print date for the above tabs only
'Rather than using .TextAlign property, text is centered here using this method
'to ensure page centering regardless of margins or paragraph settings
InitPageProperties
frmPrint.vsPrinter1.FontName = "Arial"
frmPrint.vsPrinter1.FontSize = 8
sText = "Printed: " + Date$ + " with " + App.Title + " software."
frmPrint.vsPrinter1.Measure = sText      'set string to measure
lTextHeight = frmPrint.vsPrinter1.TextHeight 'get text height
lTextWidth = frmPrint.vsPrinter1.TextWidth 'get text width
frmPrint.vsPrinter1.CurrentX = (frmPrint.vsPrinter1.PageWidth - lTextWidth) / 2

If frmPrint.vsPrinter1.CurrentY < 13000 Then
    frmPrint.vsPrinter1.CurrentY = frmPrint.vsPrinter1.PageHeight - (frmPrint.vsPrinter1.MarginBottom + (2.5 * lTextHeight)) 'set line to very bottom
Else
    frmPrint.vsPrinter1.CurrentY = frmPrint.vsPrinter1.PageHeight - (frmPrint.vsPrinter1.MarginBottom + (0.1 * lTextHeight)) 'set line to very bottom
End If

frmPrint.vsPrinter1 = sText

sText = "Copyright 1998 by SangStat Medical Corporation"
frmPrint.vsPrinter1.Measure = sText      'set string to measure
lTextWidth = frmPrint.vsPrinter1.TextWidth 'get text width
frmPrint.vsPrinter1.CurrentX = (frmPrint.vsPrinter1.PageWidth - lTextWidth) / 2
frmPrint.vsPrinter1 = sText

```

End Sub

## Public Sub PrintPageNumber()

```

'Print page number if check box is active on form
giPrintedPageNumber = giPrintedPageNumber + 1      'increment page number for next time
If gbPageNumberSuspend = False Then
    frmPrint.vsPrinter1.HdrFontSize = 8
    frmPrint.vsPrinter1.Footer = "[Dosing Report " + PAT_DATA.sPatientLastName + ", " + PAT_DATA.sPatientFirstName + "; " + PAT_DATA.sPatientID + " Page " + CStr(giPrintedPageNumber)
Else
    frmPrint.vsPrinter1.Footer = "" 'must print a blank footer otherwise old page # will show
End If

```

End Sub

## Printing.bas - RefreshPreview

54

## Public Sub RefreshPreview()

```

    Static bRefreshPreviewInProgress As Integer
    'prevent recursive calls to here
    If bRefreshPreviewInProgress = True Then Exit Sub
    If gbPreventPreviewUpdates Then Exit Sub
    bRefreshPreviewInProgress = True

    frmPrint.MousePointer = vbHourglass
    frmPrint.HScroll1.Enabled = False
    frmPrint.HScroll1.Refresh
    frmPrint.HScroll1.Value = 1
    DoEvents
    On Error GoTo 0                'reset error processing

    frmPrint.btnRefresh.Enabled = False
    frmPrint.btnRefresh.Refresh

    frmPrint.btnPrintNow.Enabled = False
    frmPrint.btnPrintNow.Refresh

    frmPrint.btnClose.Enabled = False        'disable buttons until preview build is complete
    frmPrint.btnClose.Refresh

    'frmPrint.btnFormat.Enabled = False
    'frmPrint.btnFormat.Refresh
    DoEvents

    glTotalPrintPages = 0                'reset the page counter
    glPrintedPageNumber = 1

    frmPrint.vsPrinter1.Preview = True        'print to screen
    frmPrint.vsPrinter1.Footer = ""           'Clear the footer
    'frmPrint.vsPrinter1.HdrFontName = "font name goes here" 'Controls footer also
    'frmPrint.vsPrinter1.HdrFontSize = ?? 'Controls footer also

    'Send information to the preview screen
    'Initialize print job
    InitPageMargins

    If gbPrinterErrorDetected Then GoTo RefreshPreview_Exit

    frmPrint.vsPrinter1.PreviewPage = 1        'show 1st page
    frmPrint.vsPrinter1.PreviewMode = 0        '0=screen compatible, 1=prnt compat, 2 = force monochrome
    frmPrint.vsPrinter1.PageBorder = 0        'no page border
    frmPrint.vsPrinter1.TextAlign = 0         'left align text

    'Call LoadPictureToPrinterControl(False)
    Select Case gsActiveFormName
        Case "frmPatientSummary"
        Case "frmAllPatients"
            Call PrintAllPatientsSummary
        Case "frmPatientDosingReport"
            Call PrintPatientDosingReport
    End Select

    PrintPageDate                'print date for last recipe

    frmPrint.vsPrinter1.Action = paEndDoc        'END DOC
    frmPrint.vsPrinter1.Visible = True
    'vb5 says object does not support this method frmPrint.vsPrinter1.Refresh
    Call UpdatePageButtons

```

## Printing.bas - RefreshPreview

55

```
frmPrint.HScroll1.Max = giTotalPrintPages
```

```
RefreshPreview_Exit:
```

```
frmPrint.btnClose.Enabled = True      'enable buttons
```

```
DoEvents
```

```
bRefreshPreviewInProgress = False    'allow future calls to this procedure
```

```
frmPrint.btnPrintNow.Enabled = True
```

```
frmPrint.MousePointer = vbDefault
```

```
DoEvents
```

```
End Sub
```

```
Public Sub SetPreviewSize()
```

```
Dim bHeightLimit%, fTemp As Single
```

```
frmPrint.MousePointer = vbHourglass
```

```
' frmPrint.Refresh 'a refresh of the form causes controls inside a frame to disappear
```

```
' frmPrint.vsPrinter1.Visible = False
```

```
' frmPrint.vsViewPort1.Visible = False
```

```
DoEvents
```

```
If (frmPrint.vsPrinter1.PageHeight / frmPrint.vsPrinter1.PageWidth) > (frmPrint.vsViewPort1.Height / frmPrint.vsViewPort1.Width) Then
    bHeightLimit = True
```

```
Select Case frmPrint.optZoom(0).Value
```

```
Case True 'full page view
```

```
If bHeightLimit = True Then 'there is a height restriction in the viewport control for this print orientation
```

```
frmPrint.vsPrinter1.Height = frmPrint.vsViewPort1.Height * 0.99
```

```
fTemp = frmPrint.vsPrinter1.PageWidth / frmPrint.vsPrinter1.PageHeight
```

```
fTemp = frmPrint.vsPrinter1.Height * fTemp
```

```
frmPrint.vsPrinter1.Width = fTemp
```

```
Else
```

```
frmPrint.vsPrinter1.Width = frmPrint.vsViewPort1.Width * 0.99
```

```
fTemp = frmPrint.vsPrinter1.PageHeight / frmPrint.vsPrinter1.PageWidth
```

```
fTemp = frmPrint.vsPrinter1.Width * fTemp
```

```
frmPrint.vsPrinter1.Height = fTemp
```

```
End If
```

```
'Make viewport virtual screen large enough to show full page of print control
```

```
frmPrint.vsViewPort1.VirtualWidth = frmPrint.vsPrinter1.Width * 1
```

```
frmPrint.vsViewPort1.VirtualHeight = frmPrint.vsPrinter1.Height * 1
```

```
frmPrint.vsViewPort1.BorderStyle = 1 'turn off border
```

```
Case Else 'Magnify view
```

```
frmPrint.vsPrinter1.Width = frmPrint.vsPrinter1.PageWidth * 1
```

```
frmPrint.vsPrinter1.Height = frmPrint.vsPrinter1.PageHeight * 1
```

```
frmPrint.vsViewPort1.VirtualWidth = frmPrint.vsPrinter1.Width * 1
```

```
frmPrint.vsViewPort1.VirtualHeight = frmPrint.vsPrinter1.Height * 1
```

```
frmPrint.vsViewPort1.BorderStyle = 0 'turn on border
```

```
'ensure scroll bars will be shown
```

```
End Select
```

```
frmPrint.vsPrinter1.Visible = True
```

```
frmPrint.vsViewPort1.Visible = True
```

```
frmPrint.vsViewPort1.Refresh
```

```
frmPrint.MousePointer = vbDefault
```

```
DoEvents
```

```
End Sub
```

---

Printing.bas - UpdatePageButtons

---

56

**Public Sub UpdatePageButtons()**

frmPrint.lblPageNumber.Caption = "Page " + CStr(frmPrint.HScroll1.Value) + " of " + CStr(giTotalPrintPages)

frmPrint.lblPageNumber.Refresh

If giTotalPrintPages &lt; 2 Then

frmPrint.HScroll1.Enabled = False *'no scroll bar needed for a single page*

Else

frmPrint.HScroll1.Enabled = True

End If

DoEvents

End Sub

## Fax.bas - File Declarations

57

```

Attribute VB_Name = "modFax"
Option Explicit
Public gcFax As Control
Public gsFaxFileSpec As String
Public gsEditName As String 'a temporary place to hold fax names being edited or created
Public gsEditVoice As String
Public gsEditFax As String

Public gsEditGroupIndexes As String 'holds temporary indexes to all locations associated with a group
Public gsEditGroupName As String

Type FaxDataStructure
sFaxID As String
sDialPrefix As String
iRetries As Integer
iRetryInterval As Integer
bFaxResolution As Byte
sSenderName As String
sSenderCompany As String
sSenderFaxNumber As String
sSenderVoiceNumber As String

iLocTotal As Integer 'a count of the locations
sLocPersonName(100) As String 'ugh it may be desirable in the future to make these arrays dynamic
sLocFaxNumber(100) As String
sLocVoiceNumber(100) As String

iGroupsTotal As Integer
sGroupTitle(50) As String
sGroupNamesInTitle(50) As String 'indexes to names separated by pipe. (ie 3|6|15)

iGroupLastSelected As Integer
End Type
Public FAX_DATA As FaxDataStructure

```

## Public Sub GetFaxLocations()

```
Dim l As Integer, r As Integer, sSection As String
```

```
With FAX_DATA
```

```
sSection = "Fax Locations"
```

```
.iLocTotal = CInt(GetINISetting(gsFaxFileSpec, sSection, "Total Locations", "0"))
```

```
For i = 1 To .iLocTotal
```

```
.sLocPersonName(i) = GetINISetting(gsFaxFileSpec, sSection, "Person " + CStr(i), "")
```

```
.sLocFaxNumber(i) = GetINISetting(gsFaxFileSpec, sSection, "Fax " + CStr(i), "")
```

```
.sLocVoiceNumber(i) = GetINISetting(gsFaxFileSpec, sSection, "Voice " + CStr(i), "")
```

```
Next i
```

```
sSection = "Fax Groups"
```

```
.iGroupsTotal = GetINISetting(gsFaxFileSpec, sSection, "Total Groups", "0")
```

```
For i = 0 To .iGroupsTotal
```

```
.sGroupTitle(i) = GetINISetting(gsFaxFileSpec, sSection, "Group " + CStr(i), "")
```

```
.sGroupNamesInTitle(i) = GetINISetting(gsFaxFileSpec, sSection, "Group Locations " + CStr(i), "")
```

```
Next i
```

```
sSection = "User Selections"
```

```
.iGroupLastSelected = CInt(GetINISetting(gsFaxFileSpec, sSection, "Last Group Selected", "0"))
```

```
End With
```

```
End Sub
```



## Fax.bas - GetIndexToFaxGroupName

58

**Public Function GetIndexToFaxGroupName(ByVal sGroup As String) As Integer**

*'Find sName in the list of fax names. If found, pass index back to caller.  
'otherwise return 0*

```
Dim i As Integer
sGroup = LCase$(sGroup)
With FAX_DATA
    For i = 1 To .iGroupsTotal
        If LCase$(sGroupTitle(i)) = sGroup Then
            GetIndexToFaxGroupName = i
            Exit Function
        End If
    Next i
End With
```

End Function

**Public Function GetIndexToFaxLocName(ByVal sName As String) As Integer**

*'Find sName in the list of fax names. If found, pass index back to caller.  
'otherwise return 0*

```
Dim i As Integer
sName = LCase$(sName)
With FAX_DATA
    For i = 1 To .iLocTotal
        If LCase$(sLocPersonName(i)) = sName Then
            GetIndexToFaxLocName = i
            Exit For
        End If
    Next i
End With
```

End Function

**Public Sub RemoveGroupFromFaxList(ByVal sGroup As String)**

*'Remove the name from the list and move up all others in the list.  
Dim i As Integer, j As Integer, iIndexFound As Integer*

```
With FAX_DATA
    For i = 1 To .iGroupsTotal
        If sGroupTitle(i) = sGroup Then
            iIndexFound = i
            Exit For
        End If
    Next i
```

```
For i = iIndexFound To .iGroupsTotal - 1
    .sGroupTitle(i) = .sGroupTitle(i + 1)
    .sGroupNamesInTitle(i) = .sGroupNamesInTitle(i + 1)
Next i
```

```
.iGroupsTotal = .iGroupsTotal - 1
```

End With

End Sub

## Fax.bas - RemoveNameFromFaxList

59

**Public Sub RemoveNameFromFaxList(ByVal sName As String)***'Remove the name from the list and move up all others in the list.**Dim i As Integer, j As Integer, iIndexFound As Integer, r As Integer**Dim sTempList(100) As String, sNewIndexes As String, iTemp As Integer**With FAX\_DATA**For i = 1 To .iLocTotal 'look through whole list for name**If .sLocPersonName(i) = sName Then 'found it here**iIndexFound = i**Exit For**End If**Next i**For i = iIndexFound To .iLocTotal - 1**.sLocPersonName(i) = .sLocPersonName(i + 1)**.sLocVoiceNumber(i) = .sLocVoiceNumber(i + 1)**.sLocFaxNumber(i) = .sLocFaxNumber(i + 1)**Next i**.iLocTotal = .iLocTotal - 1**'Now that the name has been removed, we must look at all of the indexes of  
'each fax group to see if an index pointer was in there. If so, it must  
'be removed. Additionally, all index greater than the one removed must be  
'decremented by one.**If iIndexFound Then**For i = 1 To .iGroupsTotal 'look at each index record in a fax group**'Parse out all of the indexes into a list for easier processing**r = ParseDelimString(.sGroupNamesInTitle(i), "|", sTempList())**sNewIndexes = ""**If r Then 'Indexes where found for this record**For j = 1 To r**'look at each item in the list to see if it equals or great than the one removed**iTemp = CInt(sTempList(j))**If iTemp = iIndexFound Then 'same index must be removed from list**'nothing to do. Don't add it to new list of indexes**ElseIf iTemp > iIndexFound Then 'higher indexes must be decremented by one.**iTemp = iTemp - 1**sNewIndexes = sNewIndexes + CStr(iTemp) + "|"**Else 'original value is OK**sNewIndexes = sNewIndexes + CStr(iTemp) + "|"**End If**Next j**End If**.sGroupNamesInTitle(i) = sNewIndexes 'store the new list of indexes back to array**Next i**End If**End With***End Sub**

Fax.bas - SetFaxDeviceLabel

60

**Public Sub SetFaxDeviceLabel()**

*'This label on the options tab displays the status of the fax device.  
'If a fax device exists, then the label displays the device, otherwise  
'it shows an appropriate message.*

**With frmOptions.lblFaxDevice****If** gcFax.DeviceCount > 0 **Then** *'at least one fax device was found**' For i = 0 To gcFax.DeviceCount - 1**.Caption = gcFax.Devices(0) 'show name of the device found**.BackColor = &HFF00& 'green background**.ForeColor = &H0&**Next i***Else** *'no fax devices were found**.Caption = "A fax device was not found. Please ensure the fax or modem is connected properly."**.BackColor = &H80& 'red background**.ForeColor = &HFFFFFF 'white***End If****End With****End Sub**

## Calendar.bas - File Declarations

61

```

Attribute VB_Name = "modCalendar"
Option Explicit

Private giCompliedDosesCreated As Integer      'number of Complied Doses to show on the calendar
Private giNonCompliedDosesCreated As Integer   'number of non-complied Doses to show on the calendar
Private giDoseSizeChangesCreated As Integer
Private giZoomDosesCreated As Integer          'number of Doses to show in zoom box
Private giDosesMissedCreated As Integer        'number of objects to show for missed days
Private gbCalendarUpdateInProgress As Integer  'prevents recursive calls while updating calendar

Public gsngComplianceTimeRange As Single      '% of hrs on either side of a prescribed dose in which a dose must be taken

Type CALENDAR_SELECTIONS
    chkDosesTaken As Byte
    chkDosesNotComplied As Byte
    chkDosesMissed As Byte
    chkDoseChanged As Byte
End Type
Public CAL_DEFAULTS As CALENDAR_SELECTIONS

Type SUMMARY_SELECTIONS
    cmbaDataToView As Byte
    cmboChartType As Byte
End Type
Public PAT_SUM_DEFAULTS As SUMMARY_SELECTIONS

```

---

**Public Function CalcDaysInMonth(ByVal iMonth As Integer, ByVal iYear As Integer)**

*'Calculate the number of days in the month/year that is passed here*

Dim i As Integer, iTemp As Long

i = iMonth + 1

If i = 13 Then i = 1

iTemp = CDate(CStr(i) + "/01/" + CStr(iYear))

iTemp = iTemp - 1

CalcDaysInMonth = Day(iTemp)

End Function

---

**Public Sub DrawAllDoseSizeChanges()**

Dim i As Integer, r As Integer, iDaysInMonth As Integer

Dim sCalendarStartDate As String, dTime As Double

Dim iDateDifference As Long, iCalendarStartDate As Long

Dim bFirstDayAlreadyPlotted As Boolean, bLastDayAlreadyPlotted As Boolean

RemoveDoseSizeChanges *'remove all of the old doses first*

iDaysInMonth = CalcDaysInMonth(frmDosingCalendar.Calendar.Month, frmDosingCalendar.Calendar.Year)

sCalendarStartDate = CStr(frmDosingCalendar.Calendar.Month) + "/01/" + Str\$(frmDosingCalendar.Calendar.Year)

iCalendarStartDate = DateValue(sCalendarStartDate)

If frmDosingCalendar.chkDoseChanged.Value Then

For i = 1 To PAT\_DATA.iEventData(0) *'total number of events*

If PAT\_DATA.byteEventType(i) = giEVENT\_DOSE\_CHANGED Then *'show only med events (not errors, etc)*

iDateDifference = Int(PAT\_DATA.dEventDate(i)) - iCalendarStartDate

If iDateDifference >= 0 And iDateDifference < iDaysInMonth Then

dTime = PAT\_DATA.dEventDate(i) - Int(PAT\_DATA.dEventDate(i))

DrawSingleDoseSizeChange CInt(iDateDifference + 1), dTime, i, True

End If

End If

Next i

End If

*'This section of code ensures that dosing info is always plotted on the first and last day of the month.*

If bFirstDayAlreadyPlotted = False Then

## Calendar.bas - DrawAllDoseSizeChanges

62

```

    r = FindPrescribedDoseSizeForSpecificDay(PAT_DATA, iCalendarStartDate)
    DrawSingleDoseSizeChange 1, dTime, r, False
End If

```

```

If bLastDayAlreadyPlotted = False Then
    r = FindPrescribedDoseSizeForSpecificDay(PAT_DATA, iCalendarStartDate + iDaysInMonth - 1)
    DrawSingleDoseSizeChange iDaysInMonth, dTime, r, False
End If

```

```

For i = 1 To giDoseSizeChangesCreated      'show all the Doses
    frmDosingCalendar.shapeDoseSizeChange(i).Visible = True
Next i

```

```
End Sub
```

## Public Sub DrawAllCompliedDosesTaken()

```

Dim i As Integer, r As Integer
Dim sCalendarStartDate As String, dTime As Double
Dim iDateDifference As Long, iCalendarStartDate As Long
Dim iDayDoseCount As Integer, iDayNumberBeingPlotted As Integer, iLastDoseDayDrawn As Integer
Dim iDaysInMonth As Integer, iTemp As Long

```

```

RemoveCompliedDosesTaken      'remove all of the old doses first
If frmDosingCalendar.chkDosesTaken.Value = False Then Exit Sub

```

```

sCalendarStartDate = CStr(frmDosingCalendar.Calendar.Month) + "/01/" + Str$(frmDosingCalendar.Calendar.Year)
iCalendarStartDate = DateValue(sCalendarStartDate)

```

```

'Calc the number of days in the month being displayed
iDaysInMonth = CalcDaysInMonth(frmDosingCalendar.Calendar.Month, frmDosingCalendar.Calendar.Year)

```

```

For i = 1 To PAT_DATA.iEventData(0)      'total number of events
    If PAT_DATA.byteEventType(i) = giEVENT_DOSE_TAKEN Then      'show only med events (not errors, etc)
        iDateDifference = Int(PAT_DATA.dEventDate(i)) - iCalendarStartDate
        If iDateDifference >= 0 And iDateDifference < iDaysInMonth Then      'dose occurred during this month
            'Determine if the dose occurred within the compliance parameters
            r = IsDoseWithinPrescribedTimeRange(PAT_DATA, i)      'pass index to event time
            If r Then
                iDayNumberBeingPlotted = CInt(iDateDifference + 1)
                If iDayNumberBeingPlotted = iLastDoseDayDrawn Then
                    iDayDoseCount = iDayDoseCount + 1      'plotting same day as last dose
                Else
                    iDayDoseCount = 1      'this is a new day. Reset counter
                End If
                iLastDoseDayDrawn = iDayNumberBeingPlotted      'remember that we are plotting on this day
                DrawSingleCompliedDoseTaken iDayNumberBeingPlotted, dTime, i, iDayDoseCount
            End If
        End If
    End If
Next i

```

```

For i = 1 To giCompliedDosesCreated      'show all the Doses
    frmDosingCalendar.shapeDose(i).Visible = True
Next i

```

```
End Sub
```

## Calendar.bas - DrawAllDosesMissed

63

**Public Sub DrawAllDosesMissed()**

```

Dim i As Integer, iDaysInMonth As Integer, i As Long
Dim sCalendarStartDate As String
Dim iDateDifference As Long, iCalendarStartDate As Long
Dim iDayDoseCount As Integer, iDayBeingPlotted As Integer

```

```

RemoveDosesMissed 'remove all of the old doses first
If frmDosingCalendar.chkDosesMissed.Value = False Then Exit Sub

```

```

iDaysInMonth = CalcDaysInMonth(frmDosingCalendar.Calendar.Month, frmDosingCalendar.Calendar.Year)
sCalendarStartDate = CStr(frmDosingCalendar.Calendar.Month) + "/01/" + CStr(frmDosingCalendar.Calendar.Year)
iCalendarStartDate = DateValue(sCalendarStartDate)

```

```

For i = iCalendarStartDate To iCalendarStartDate + iDaysInMonth - 1 'sequence through all days in month
    If i >= PAT_DATA.dEventDate(1) Then 'day being plotted is not earlier than 1st dose in structure
        If i < PAT_DATA.dEventDate(PAT_DATA.iEventData(0)) Then 'day being plotted is not later than last dose in structure
            iDayBeingPlotted = i - iCalendarStartDate + 1 'get the current month day to plot
            iDayDoseCount = CalcDosesSumTakenOnSpecificDay(PAT_DATA, i) 'calc missed doses for this day
            For i = 1 To PAT_DATA.iDosesPerDay - iDayDoseCount
                DrawSingleDoseMissed iDayBeingPlotted, i 'Plot the current day
            Next i
        End If
    End If
Next i

```

```

For i = 1 To giDosesMissedCreated 'show all the Doses
    frmDosingCalendar.shapeDoseMissed(i).Visible = True
Next i

```

```

End Sub

```

**Public Sub DrawAllNonCompliedDosesTaken()**

```

Dim i As Integer, j As Integer, bDoseOutOfCompliance As Boolean
Dim dTimeLimit As Double, dLowLimit As Double, dHighLimit As Double
Dim sCalendarStartDate As String, dTime As Double
Dim iDateDifference As Long, iCalendarStartDate As Long
Dim iDayDoseCount As Integer, iDayNumberBeingPlotted As Integer, iLastDoseDayDrawn As Integer
Dim iDaysInMonth As Integer

```

```

RemoveNonCompliedDosesTaken 'remove all of the old doses first
If frmDosingCalendar.chkDosesNotComplied.Value = False Then Exit Sub

```

```

iDaysInMonth = CalcDaysInMonth(frmDosingCalendar.Calendar.Month, frmDosingCalendar.Calendar.Year)
sCalendarStartDate = CStr(frmDosingCalendar.Calendar.Month) + "/01/" + Str$(frmDosingCalendar.Calendar.Year)
iCalendarStartDate = DateValue(sCalendarStartDate)

```

```

dTimeLimit = gsngComplianceTimeRange / 24
For i = 1 To PAT_DATA.iEventData(0) 'total number of events
    If PAT_DATA.byteEventType(i) = giEVENT_DOSE_TAKEN Then 'show only med events (not errors, etc)
        iDateDifference = Int(PAT_DATA.dEventDate(i)) - iCalendarStartDate
        If iDateDifference >= 0 And iDateDifference <= iDaysInMonth Then 'dose occurred during this month
            dTime = PAT_DATA.dEventDate(i) - Int(PAT_DATA.dEventDate(i)) 'get time of dose
            'Determine if the dose occurred within the compliance parameters
            'rgh see if we can use our procedure already created

            If gsngComplianceTimeRange Then 'do test if there is a value set in the compliance time range
                bDoseOutOfCompliance = True 'set default to be out of range unless otherwise set below
                For j = 1 To PAT_DATA.iDosesPerDay
                    'compare dose time against all of the alarm times
                    dLowLimit = PAT_DATA.dPrescribedDoseTime(j) - dTimeLimit - 0.0001 'add a factor to prevent rounding error
                    dHighLimit = PAT_DATA.dPrescribedDoseTime(j) + dTimeLimit + 0.0001
                    If dTime >= dLowLimit And dTime <= dHighLimit Then 'this dose is within compliance
                        bDoseOutOfCompliance = False 'set flag to not plot this dose
                        Exit For 'no need for further testing of this dose. It is in compliance
                    End If
                Next j
            End If
        End If
    End If
Next i

```

## Calendar.bas - DrawAllNonCompliedDosesT{

64

```

Else 'there is no compliance range
    bDoseOutOfCompliance = False
End If

If bDoseOutOfCompliance Then
    iDayNumberBeingPlotted = CInt(IDateDifference + 1)
    If iDayNumberBeingPlotted = iLastDoseDayDrawn Then
        iDayDoseCount = iDayDoseCount + 1 'plotting same day as last dose
    Else
        iDayDoseCount = 1 'this is a new day. Reset counter
    End If
    iLastDoseDayDrawn = iDayNumberBeingPlotted 'remember that we are plotting on this dat
    DrawSingleNonCompliedDoseTaken CInt(IDateDifference + 1), dTime, i, iDayDoseCount
End If
End If
End If
Next i

For i = 1 To giNonCompliedDosesCreated 'show all the Doses
    frmDosingCalendar.shapeDoseNonComply(i).Visible = True
Next i
End Sub

```

**Public Sub DrawSingleDoseSizeChange(iDay As Integer, dTime As Double, iEventNumber As Integer, bHighlight**  
*'Draw dosing Doses for the day of the month and time of day passed in here.*

*'Time of time is expressed in decimal places as a portion of a day (VB time format)*  
*'NOTE: No checks are currently made to determine whether the event is a med event or*  
*'a non-med event. If both events are kept in the same array, then a test of the med bit*  
*'must be done before plotting.*  
*'A Dose is not visible when first created. The caller should display the Doses once they*  
*'are all created, so as to speed the redraw of the screen.*  
*'Create another clone of the the Dose shape located in the array Dose(0)*  
*'When this feature is on, a dose size is automatically entered on the first and last day of the month.*

*' On Error Resume Next*

*Dim I As Integer, iWeekDay As Integer*  
*Dim iDoseLeft As Single, iDoseTop As Single*  
*Dim iTemp As Long, iDayWidth As Long, iDayHeight As Long*

*giDoseSizeChangesCreated = giDoseSizeChangesCreated + 1 'increment counter*  
*Load frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated) 'create a new object*  
*DoEvents*  
*frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated) = " " + CStr(PAT\_DATA.iEventData(iEventNumber)) + " mg"*

*If bHighlight Then*  
*frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).BackColor = &HFFFFFF00 'blue*  
*frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).ToolTipText = " Dose Size Was Changed Today "*  
*Else*  
*frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).ToolTipText = " Current Dose Size "*  
*End If*

*'These lines are a work-around for a bug in the control that causes it to*  
*'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties*  
*'and remove these calculations.*  
*iDayWidth = (frmDosingCalendar.Calendar.Width - 50) / 7 'actual scalewidth of a single day*  
*iTemp = (frmDosingCalendar.Calendar.DayLeft(iDay) \* 25) / iDayWidth 'approximate location of the day*  
*iWeekDay = CInt(iTemp)*

*iDoseLeft = (iWeekDay \* iDayWidth) 'get left edge of day to plot*  
*'iDoseLeft = iDoseLeft + ((iDayWidth / 5) \* iPlotPosition - 1) - (iDayWidth / 10)*  
*iDoseLeft = iDoseLeft + (iDayWidth \* 0.8) - frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).Width*

*If Int(iDayWidth / 150) < 7 Then*  
*frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).FontBold = False*  
*Else*

## Calendar.bas - DrawSingleDoseSizeChang

65

```
frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).FontBold = True
End If
```

```
frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).FontSize = Int(IDayWidth / 150)
frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).Left = IDoseLeft
```

*'These lines are a work-around for a bug in the control that causes it to  
'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties  
'and remove these calculations.  
'The control is even more stupid than I first suspected. It can not always return the proper vertical  
'location of a day, thus, we have to jump through more hoops to figure out what week that a particular  
'day is in.*

```
IDayHeight = (frmDosingCalendar.Calendar.Height - 50) - 625      'an offset is used to compensate for height of title
IDayHeight = IDayHeight / 6
iWeekDay = (frmDosingCalendar.Calendar.DayLeft(1) * 26) / IDayWidth    'the approximate location of the day
iTemp = Int((IDay + iWeekDay - 1) / 7)
```

```
IDoseTop = (iTemp * IDayHeight) + 625    'this number factors in the title bar
IDoseTop = IDoseTop + 50
```

```
frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).Top = IDoseTop
frmDosingCalendar.shapeDoseSizeChange(giDoseSizeChangesCreated).Tag = iEventNumber
```

*'keep event number for updating the  
zoom: box*

*On Error GoTo 0*

End Sub

### Private Sub DrawSingleNonCompliedDoseTaken(iDay As Integer, dTime As Double, iEventNumber As Integer, iPlotPosition As Integer)

*'Draw dosing Doses for the day of the month and time of day passed in here.  
'Time of time is expressed in decimal places as a portion of a day (VB time format)  
'NOTE: No checks are currently made to determine whether the event is a med event or  
'a non-med event. If both events are kept in the same array, then a test of the med bit  
'must be done before plotting.*

*'A Dose is not visible when first created. The caller should display the Doses once they  
'are all created, so as to speed the redraw of the screen.*

*'Create another clone of the the Dose shape located in the array Dose(0)*

*On Error Resume Next*

*Dim i As Integer, iWeekDay As Integer*

*Dim IDoseLeft As Single, IDoseTop As Single*

*Dim iTemp As Long, IDayWidth As Long, IDayHeight As Long*

```
giNonCompliedDosesCreated = giNonCompliedDosesCreated + 1    'increment Dose counter
```

```
Load frmDosingCalendar.shapeDoseNonComply(giNonCompliedDosesCreated)    'create a new Dose
```

*'These lines are a work-around for a bug in the control that causes it to  
'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties  
'and remove these calculations.*

```
IDayWidth = (frmDosingCalendar.Calendar.Width - 50) / 7      'actual scalewidth of a single day
```

```
iWeekDay = (frmDosingCalendar.Calendar.DayLeft(iDay) * 26) / IDayWidth    'approximate location of the day
```

```
IDoseLeft = (iWeekDay * IDayWidth)    'get left edge of day to plot
```

```
IDoseLeft = IDoseLeft + ((IDayWidth / 5) * iPlotPosition - 1) - (IDayWidth / 10)
```

```
frmDosingCalendar.shapeDoseNonComply(giNonCompliedDosesCreated).Left = IDoseLeft
```

*'These lines are a work-around for a bug in the control that causes it to  
'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties  
'and remove these calculations.*

*'The control is even more stupid than I first suspected. It can not always return the proper vertical  
'location of a day, thus, we have to jump through more hoops to figure out what week that a particular  
'day is in.*

```
IDayHeight = (frmDosingCalendar.Calendar.Height - 50) - 625    'an offset is used to compensate for height of title
```



## Calendar.bas - DrawSingleNonCompliedDose

66

```

IDayHeight = IDayHeight / 6
iWeekDay = (frmDosingCalendar.Calendar.DayLeft(1) * 26) / IDayWidth      'the approximate location of the day
ITemp = Int((IDay + iWeekDay - 1) / 7)

IDoseTop = (ITemp * IDayHeight) + 625      'this number factors in the title bar
IDoseTop = IDoseTop + IDayHeight - 50 - frmDosingCalendar.shapeDose(0).Height - frmDosingCalendar.shapeDoseNonComply(0).Height      'draw in middle of day
frmDosingCalendar.shapeDoseNonComply(giNonCompliedDosesCreated).Top = IDoseTop

frmDosingCalendar.shapeDoseNonComply(giNonCompliedDosesCreated).Tag = iEventNumber      'keep event number for updating the zoom box

On Error GoTo 0
End Sub

```

**Public Function IsDoseWithinPrescribedTimeRange(DataStruct As DeviceDataStruct, ByVal iIndex As Integer)**

*'Test to see that the event at the index passed here is a medication event and that it is within the prescribed time range for a daily dose. If yes, then pass TRUE back to the caller.*

*Dim i As Integer, dTime As Double*  
*Dim dTimeLimit As Double, dLowLimit As Double, dHighLimit As Double*

```

dTimeLimit = gsngComplianceTimeRange / 24
dTime = DataStruct.dEventDate(iIndex) - Int(DataStruct.dEventDate(iIndex))      'get time of dose
If gsngComplianceTimeRange Then
    For i = 1 To DataStruct.iDosesPerDay
        'compare dose time against all of the alarm times
        dLowLimit = DataStruct.dPrescribedDoseTime(i) - dTimeLimit - 0.0001      'add a factor to prevent rounding error
        dHighLimit = DataStruct.dPrescribedDoseTime(i) + dTimeLimit + 0.0001
        If dTime >= dLowLimit And dTime <= dHighLimit Then
            IsDoseWithinPrescribedTimeRange = True
            Exit For      'no need to do any further testing for this dose
        End If
    Next i
Else      'there is no compliance range, so pass back a success flag
    IsDoseWithinPrescribedTimeRange = True
End If
End Function

```

**Private Sub PrintCalendar()**

*" This routine is called when the user presses the print button on the calendar form*

```

' Dim sPrintInfo As String
' Dim CRLF As String
' Dim bcolorCalendar As Long
' Dim bcolorpnlZoom As Long
' Dim bcolorpnlTime As Long
' Dim bcolorForm As Long
' Dim fcolorPrescribed As Long
' Dim fcolorMissed As Long
' Dim fcolorWeek As Long

' Dim curTop As Long
' Dim curWidth As Integer
' Dim curHeight As Integer

' Const XOffset = 1920
' Const YOffset = 1890

' On Error GoTo Error_btnPrint

CRLF = Chr$(13) + Chr$(10)

```

## Calendar.bas - PrintCalendar

67

```

*
* curTop = Me.Top
* curWidth = Me.Width
* curHeight = Me.Height
*
* Hide this guy off the screen while we print
* Me.Top = -(curHeight * 2)
*
* Save current background colors
* bcolorCalendar = Calendar.BackColor
* bcolorpniZoom = pniZoom.BackColor
* bcolorpniTime = pniTime.BackColor
* bcolorForm = Me.BackColor
* fcolorPrescribed = chkDosesTaken.FillColor
* fcolorMissed = chkDosesMissed.FillColor
* fcolorvWeek = chkVWeekNumbers.FillColor
*
* hide the buttons
* btnClose.Visible = False
* btnPrint.Visible = False
*
* Add date + time info to printed data
* sPrintInfo = "Printed on: " + Format$(Now, "dddd hh:nn")
* lblPrintInfo.Caption = sPrintInfo
*
* Set titles at top of printed page
* lblTitle.Caption = "Dosing Calendar"
* lblPatient.Caption = "Patient: " + tgDeviceStat.sPatient
* lblDrug.Caption = "Drug: " + tgDeviceStat.sDrug
*
* Set background colors
* Calendar.BackColor = WHITE
* pniZoom.BackColor = WHITE
* pniTime.BackColor = WHITE
* Me.BackColor = WHITE
* chkDosesTaken.FillColor = WHITE
* chkDosesMissed.FillColor = WHITE
* chkWeekNumbers.FillColor = WHITE
*
* Let user know we are printing
* Load frm_Status
* frm_Status.lblStatus.Caption = "Preparing to print calendar"
* frm_Status.Show
*
* Move resize the form and move objects to give space for printing
* Call DeleteAllObjects ' remove extraneous elements from calendar
* Call MoveFormObjects(Me, XOffset, YOffset, True)
* Call UpdateCalendar
* DoEvents
*
* Call UpdateZoomBox
*
* Switch on visibility of titles
* lblPrintInfo.Visible = True
* lblTitle.Visible = True
* lblPatient.Visible = True
* lblDrug.Visible = True
*
* Make Check boxes two dimensional
* chkDosesTaken.CheckBox2d = True
* chkDosesMissed.CheckBox2d = True
* chkWeekNumbers.CheckBox2d = True
*
* Bring the Zoom labels to the front
* lblZoomTime.ZOrder 0
* lblZoomTime.Visible = True
*
* Print the form

```

## Calendar.bas - PrintCalendar

68

```
.  
· Me.Height = Me.Height - YOffset  
· Me.Width = Me.Width + XOffset  
·  
· DoEvents  
· Me.PrintForm  
· DoEvents  
·  
· frm_Status.lblStatus.Caption = "Sending calendar to printer"  
·  
· ' hide the titles and show the buttons  
· lblPrintInfo.Visible = False  
· lblTitle.Visible = False
```

## Calendar.bas - PrintCalendar

69

```

* !blPatient.Visible = False
* !blDrug.Visible = False
*
* * Move everything back
* Call DeleteAllObjects
* Call MoveFormObjects(Me, -XOffset, -YOffset, True)
* Call UpdateCalendar
*
* * Restore background colors
* Calendar.BackColor = bcolorCalendar
* pnlZoom.BackColor = bcolorpnlZoom
* pnlTime.BackColor = bcolorpnlTime
* Me.BackColor = bcolorForm
* chkDosesTaken.FillColor = fcolorPrescribed
* chkDosesMissed.FillColor = fcolorMissed
* chkWeekNumbers.FillColor = fcolorWeek
*
* * Restore buttons
* btnClose.Visible = True
* btnPrint.Visible = True
*
* * Set check boxes back to 3d
* * Make Check boxes two dimensional
* chkDosesTaken.CheckBox2d = False
* chkDosesMissed.CheckBox2d = False
* chkWeekNumbers.CheckBox2d = False
*
* * Bring box back into view
* Me.Width = curWidth
* Me.Height = curHeight
* Me.Top = curTop      ' bring form back into view
*
* Unload frm_Status
*
*Exit_btnPrint:
* Exit Sub
*
*
*Error_btnPrint:
* Resume Exit_btnPrint

```

## Calendar.bas - PrintCalendar

70

---

End Sub

---

**Public Sub RemoveDoseSizeChanges()**

Dim i As Integer  
On Error Resume Next

For i = 1 To giDoseSizeChangesCreated     *'remove all previous Doses*  
    Unload frmDosingCalendar.shapeDoseSizeChange(i)  
Next i  
giDoseSizeChangesCreated = 0

On Error GoTo 0  
End Sub

---

**Public Sub RemoveDosesMissed()**

Dim i As Integer  
On Error Resume Next

For i = 1 To giDosesMissedCreated     *'remove all previous objects*  
    Unload frmDosingCalendar.shapeDoseMissed(i)  
Next i  
giDosesMissedCreated = 0

On Error GoTo 0  
End Sub

---

**Public Sub RemoveCompliedDosesTaken()**

Dim i As Integer  
On Error Resume Next

For i = 1 To giCompliedDosesCreated     *'remove all previous Doses*  
    Unload frmDosingCalendar.shapeDose(i)  
Next i  
giCompliedDosesCreated = 0

On Error GoTo 0  
End Sub

---

**Public Sub RemoveNonCompliedDosesTaken()**

Dim i As Integer  
On Error Resume Next

For i = 1 To giNonCompliedDosesCreated     *'remove all previous Doses*  
    Unload frmDosingCalendar.shapeDoseNonComply(i)  
Next i  
giNonCompliedDosesCreated = 0

On Error GoTo 0  
End Sub

## Calendar.bas - UpdateZoomBox

71

## Public Sub UpdateZoomBox()

*'A different day was clicked on the calendar, so we need to plot the events for the current day into the zoom box.*  
*'This procedure draws doses taken for a given day*  
*'NOTE: No checks are currently made to determine whether the event is a med event or a non-med event. If both events are kept in the same array, then a test of the med bit must be done before plotting.*

Dim i As Integer, dTime As Double, iDoseDay As Integer, iZoomPaneWidth As Integer  
 Static bProcedureInProgress  
 If bProcedureInProgress Then Exit Sub  
 bProcedureInProgress = True

On Error Resume Next *'prevent error if already unloaded*  
 For i = 1 To giZoomDosesCreated *'remove all previous Doses*  
     Unload frmDosingCalendar.shapeZoomDose(i)  
 Next i  
 giZoomDosesCreated = 0

For i = 1 To 4  
     Unload frmDosingCalendar.shapeZoomPrescribed(i) *'clear the text box for zoom time*  
     Unload frmDosingCalendar.shapeZoomTimeRange(i) *'clear the text box for zoom time*  
 Next i

On Error GoTo 0 *'resume normal error status*

iZoomPaneWidth = frmDosingCalendar.pnlZoom.Width *'speed up process by defining width from control*

For i = 1 To 4  
     If PAT\_DATA.dPrescribedDoseTime(i) >= 0 Then  
         dTime = PAT\_DATA.dPrescribedDoseTime(i) - Int(PAT\_DATA.dPrescribedDoseTime(i))  
         Load frmDosingCalendar.shapeZoomTimeRange(i)  
         frmDosingCalendar.shapeZoomTimeRange(i).Left = (iZoomPaneWidth \* dTime) - (iZoomPaneWidth \* (gsngComplianceTimeRange / 24))  
         frmDosingCalendar.shapeZoomTimeRange(i).Width = (iZoomPaneWidth \* (gsngComplianceTimeRange / 24) \* 2)  
         frmDosingCalendar.shapeZoomTimeRange(i).ToolTipText = "Compliance Time Range = +- " & CStr(gsngComplianceTimeRange) & " Hrs."  
         frmDosingCalendar.shapeZoomTimeRange(i).Visible = True  
         frmDosingCalendar.shapeZoomTimeRange(i).ZOrder  
     End If  
 Next i

For i = 1 To 4  
     If PAT\_DATA.dPrescribedDoseTime(i) >= 0 Then  
         dTime = PAT\_DATA.dPrescribedDoseTime(i) - Int(PAT\_DATA.dPrescribedDoseTime(i))  
         Load frmDosingCalendar.shapeZoomPrescribed(i)  
         frmDosingCalendar.shapeZoomPrescribed(i).Left = (iZoomPaneWidth \* dTime) - (frmDosingCalendar.shapeZoomPrescribed(i).Width / 2) + 15  
         frmDosingCalendar.shapeZoomPrescribed(i).ToolTipText = Format\$(dTime, gsTimeDisplayFormat)  
         frmDosingCalendar.shapeZoomPrescribed(i).Visible = True  
         frmDosingCalendar.shapeZoomPrescribed(i).ZOrder  
     End If  
 Next i

For i = 1 To 4  
     frmDosingCalendar.txtZoomTime(i).Caption = "" *'clear the text box for zoom time*  
 Next i

Dim iCalendarDate As Long  
 iCalendarDate = DateValue(frmDosingCalendar.Calendar.Date)  
 For i = 1 To giCompliedDosesCreated  
     *'ugh we may later want to use a global array instead of the tag property to prevent flashing and speed things up.*  
     iDoseDay = frmDosingCalendar.shapeDose(i).Tag *'get the day that the dose was taken on*  
     If Int(PAT\_DATA.dEventDate(iDoseDay)) = iCalendarDate Then  
         *'Create another clone of the the Dose shape located in the array Dose(0)*  
         giZoomDosesCreated = giZoomDosesCreated + 1 *'increment Dose counter*

## Calendar.bas - UpdateZoomBox

72

```

Load frmDosingCalendar.shapeZoomDose(giZoomDosesCreated) 'create a new Dose
dTime = PAT_DATA.dEventDate(iDoseDay) - Int(PAT_DATA.dEventDate(iDoseDay))
frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).Left = (frmDosingCalendar.pnlZoom.Width * dTime) - (
frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).Width / 2)
frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).ToolTipText = Format$(dTime, gsTimeDisplayFormat)
frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).Visible = True
frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).ZOrder

End If
Next i

For i = 1 To giNonCompliedDosesCreated
    iDoseDay = frmDosingCalendar.shapeDoseNonComply(i).Tag 'get the day that the dose was taken on
    If Int(PAT_DATA.dEventDate(iDoseDay)) = iCalendarDate Then
        'Create another clone of the the Dose shape located in the array Dose(0)
        giZoomDosesCreated = giZoomDosesCreated + 1 'increment Dose counter
        Load frmDosingCalendar.shapeZoomDose(giZoomDosesCreated) 'create a new Dose
        dTime = PAT_DATA.dEventDate(iDoseDay) - Int(PAT_DATA.dEventDate(iDoseDay))
        frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).Left = (frmDosingCalendar.pnlZoom.Width * dTime) - (
        frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).Width / 2)
        frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).ToolTipText = Format$(dTime, gsTimeDisplayFormat)
        frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).Visible = True
        frmDosingCalendar.shapeZoomDose(giZoomDosesCreated).ZOrder
    End If
Next i

frmDosingCalendar.pnlZoom.Caption = Format$(frmDosingCalendar.Calendar.Date, "General Date") + " Detail View"

'update position of time scale
For i = 2 To 22 Step 2
    frmDosingCalendar.lblDetailTime(i).Left = (frmDosingCalendar.pnlZoom.Width * (i / 24)) - (frmDosingCalendar.lblDetailTime(i).Width
    / 2)
Next i
frmDosingCalendar.shapeDayLight(2).Width = frmDosingCalendar.pnlZoom.Width * 0.53
frmDosingCalendar.shapeDayLight(1).Width = frmDosingCalendar.pnlZoom.Width * 0.03
frmDosingCalendar.shapeDayLight(3).Width = frmDosingCalendar.shapeDayLight(1).Width
frmDosingCalendar.shapeDayLight(2).Left = (frmDosingCalendar.pnlZoom.Width - frmDosingCalendar.shapeDayLight(2).Width) / 1.8
frmDosingCalendar.shapeDayLight(1).Left = 20 + frmDosingCalendar.shapeDayLight(2).Left - frmDosingCalendar.shapeDayLight(1).
Width
frmDosingCalendar.shapeDayLight(3).Left = frmDosingCalendar.shapeDayLight(2).Left + frmDosingCalendar.shapeDayLight(2).Width
- 15

bProcedureInProgress = False
End Sub

```

## Private Sub MoveFormObjects(frm As Form, XOffset As Integer, YOffset As Integer, VisibleOnly As Integer)

\* This routine moves all objects on a form by the specified amount  
 \* Argument Description  
 \* frm Form object  
 \* XOffset offset (in twips) to move in x plane. Positive is to the right  
 \* YOffset offset (in twips) to move in y plane. Positive is down.  
 \* VisibleOnly If true only move visible objects

Dim i As Integer  
 On Error GoTo Error\_MoveFormObjects

\* loop through all the forms on the form

```

For i = 0 To frm.Controls.Count - 1
    ' if 1) the processing only visible controls and the controls is visible
    ' or 2) processing all controls
    If frm.Controls(i).Tag <> "contained" Then
        If (VisibleOnly And frm.Controls(i).Visible) Or Not VisibleOnly Then
            ' reset left and top properties
            frm.Controls(i).Left = frm.Controls(i).Left + XOffset
            frm.Controls(i).Top = frm.Controls(i).Top + YOffset
        End If
    End If
Next i

```

## Calendar.bas - MoveFormObjects

73

Exit\_MoveFormObjects:  
Exit Sub

Error\_MoveFormObjects:  
Resume Exit\_MoveFormObjects

End Sub

Private Sub DrawSingleCompliedDoseTaken(iDay As Integer, dTime As Double, iEventNumber As Integer, iPlotPosition As Integer)

*'Draw dosing Doses for the day of the month and time of day passed in here.*

*'Time is expressed in decimal places as a portion of a day (V5 time format)*

*'NOTE: No checks are currently made to determine whether the event is a med event or*

*'a non-med event. If both events are kept in the same array, then a test of the med bit*

*'must be done before plotting.*

*'A Dose is not visible when first created. The caller should display the Doses once they*

*'are all created, so as to speed the redraw of the screen.*

*'Create another clone of the the Dose shape located in the array Dose(0)*

*On Error Resume Next*

*Dim i As Integer, iWeekDay As Integer*

*Dim iDoseLeft As Single, iDoseTop As Single*

*Dim iTemp As Long, iDayWidth As Long, iDayHeight As Long*

*giCompliedDosesCreated = giCompliedDosesCreated + 1*

*'increment Dose counter*

*Load frmDosingCalendar.shapeDose(giCompliedDosesCreated)*

*'create a new Dose*

*'These lines are a work-around for a bug in the control that causes it to*

*'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties*

*'and remove these calculations.*

*iDayWidth = (frmDosingCalendar.Calendar.Width - 50) / 7*

*'actual scalewidth of a single day*

*iWeekDay = (frmDosingCalendar.Calendar.DayLeft(iDay) \* 26) / iDayWidth*

*'approximate location of the day*

*iDoseLeft = (iWeekDay \* iDayWidth)*

*'get left edge of day to plot*

*iDoseLeft = iDoseLeft + ((iDayWidth / 5) \* iPlotPosition - 1) - (iDayWidth / 10)*

*frmDosingCalendar.shapeDose(giCompliedDosesCreated).Left = iDoseLeft*

*'These lines are a work-around for a bug in the control that causes it to*

*'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties*

*'and remove these calculations.*

*'The control is even more stupid than I first suspected. It can not always return the proper vertical*

*'location of a day, thus, we have to jump through more hoops to figure out what week that a particular*

*'day is in.*

*iDayHeight = (frmDosingCalendar.Calendar.Height - 50) - 625*

*'an offset is used to compensate for height of title*

*iDayHeight = iDayHeight / 6*

*iWeekDay = (frmDosingCalendar.Calendar.DayLeft(1) \* 26) / iDayWidth*

*'the approximate location of the day*

*iTemp = Int((iDay + iWeekDay - 1) / 7)*

*iDoseTop = (iTemp \* iDayHeight) + 625*

*'this number factors in the title bar*

*iDoseTop = iDoseTop + iDayHeight - 25 - frmDosingCalendar.shapeDose(0).Height*

*'draw in bottom of day*

*frmDosingCalendar.shapeDose(giCompliedDosesCreated).Top = iDoseTop*

*frmDosingCalendar.shapeDose(giCompliedDosesCreated).Tag = iEventNumber*

*'keep event number for updating the zoom box*

*On Error GoTo 0*

End Sub



## Calendar.bas - DrawSingleDoseMissed

74

**Private Sub DrawSingleDoseMissed(iDay As Integer, iPlotPosition As Integer)***'Draw doses for the day of the month passed in here**'NOTE: No checks are currently made to determine whether the event is a med event or a non-med event. A test of the med bit must be done before calling this procedure.**'A Dose is not visible when first created. The caller should display the Doses once they are all created, so as to speed the redraw of the screen.**On Error Resume Next**Dim i As Integer, iWeekDay As Integer**Dim iDoseLeft As Single, iDoseTop As Single**Dim iTemp As Long, iDayWidth As Long, iDayHeight As Long**giDosesMissedCreated = giDosesMissedCreated + 1**'increment Dose counter**'Create another clone of the the Dose shape located in the array Dose(0)**Load frmDosingCalendar.shapeDoseMissed(giDosesMissedCreated)**'create a new Dose**'These lines are a work-around for a bug in the control that causes it to**'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties**'and remove these calculations.**iDayWidth = (frmDosingCalendar.Calendar.Width - 50) / 7**'actual scalewidth of a single day**iWeekDay = (frmDosingCalendar.Calendar.DayLeft(iDay) \* 26) / iDayWidth**'approximate location of the day**iDoseLeft = (iWeekDay \* iDayWidth)**'get left edge of day to plot**iDoseLeft = iDoseLeft + ((iDayWidth / 5) \* iPlotPosition - 1) - (iDayWidth / 10)**frmDosingCalendar.shapeDoseMissed(giDosesMissedCreated).Left = iDoseLeft**'These lines are a work-around for a bug in the control that causes it to**'return the wrong values for day.left, day.top, etc. When fixed, we can simply use those properties**'and remove these calculations.**'The control is even more stupid than I first suspected. It can not always return the proper vertical**'location of a day, thus, we have to jump through more hoops to figure out what week that a particular**'day is in.**iDayHeight = (frmDosingCalendar.Calendar.Height - 50) - 625**'an offset is used to compensate for height of title**iDayHeight = iDayHeight / 6**iWeekDay = (frmDosingCalendar.Calendar.DayLeft(1) \* 26) / iDayWidth**'the approximate location of the day**iTemp = Int((iDay + iWeekDay - 1) / 7)**iDoseTop = (iTemp \* iDayHeight) + 625**'this number factors in the title bar**iDoseTop = iDoseTop + iDayHeight - 75 - frmDosingCalendar.shapeDose(0).Height - frmDosingCalendar.shapeDose(0).Height -**frmDosingCalendar.shapeDose(0).Height**'draw in bottom of day**frmDosingCalendar.shapeDoseMissed(giDosesMissedCreated).Top = iDoseTop**On Error GoTo 0**End Sub***Public Sub UpdateCalendar()***'The month or year to the calendar has changed, so we need to plot the events for the current month and year being shown.**Static bProcedureInProgress As Boolean**If bProcedureInProgress Then Exit Sub**bProcedureInProgress = True**Dim iObjectDiameter As Integer**'Show custom labels from config file if there were any**If Len(gsCustomLblPatientLastName) > 0 Then frmDosingCalendar.Label1 = gsCustomLblPatientLastName**frmDosingCalendar.lblPatientName = "" + PAT\_DATA.sPatientLastName + ", " + PAT\_DATA.sPatientFirstName**iObjectDiameter = frmDosingCalendar.Calendar.Width / 45**'resize the objects drawn on the calendar**If iObjectDiameter > frmDosingCalendar.Calendar.Height / 50 Then iObjectDiameter = frmDosingCalendar.Calendar.Height / 50*

## Calendar.bas - UpdateCalendar

75

```
frmDosingCalendar.shapeDose(0).Width = iObjectDiameter
frmDosingCalendar.shapeDose(0).Height = iObjectDiameter
frmDosingCalendar.shapeDoseNonComply(0).Width = iObjectDiameter
frmDosingCalendar.shapeDoseNonComply(0).Height = iObjectDiameter
frmDosingCalendar.shapeDoseMissed(0).Width = iObjectDiameter
frmDosingCalendar.shapeDoseMissed(0).Height = iObjectDiameter
frmDosingCalendar.shapeDoseSizeChange(0).Width = iObjectDiameter
frmDosingCalendar.shapeDoseSizeChange(0).Height = iObjectDiameter
```

```
DrawAllCompliedDosesTaken
DrawAllNonCompliedDosesTaken
DrawAllDosesMissed
DrawAllDoseSizeChanges
```

```
UpdateZoomBox
```

```
bProcedureInProgress = False
End Sub
```

---

**Public Sub RemoveAllObjects()**

```
'Remove all objects from calendar
RemoveCompliedDosesTaken
RemoveDosesMissed
DoEvents
End Sub
```

## frmMain.frm - File Declarations

76

```

Attribute VB_Name = "frmMain"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub CommTimer_Timer()
    gbCommTimerExpired = True
    CommTimer.Enabled = False
End Sub

```

```

Private Sub FaxMan1_ConfigurationDone()
    Dim i As Integer
    frmOptions.MousePointer = vbDefault
    frmOptions.btnConfigureFax.Enabled = True
    SetFaxDeviceLabel
End Sub

```

```

Private Sub FaxMan1_FaxStatus(Device As Integer, Status As Integer)
    Beep
    If gcFax.Status(Device) = "Initializing Modem" Or gcFax.Status(Device) = "Answering" Then
        frmFaxStatus.Show
    ElseIf gcFax.Status(Device) = "Port Closed" Then
        Unload frmFaxStatus
    End If

```

```

    frmFaxStatus.lblRemoteID = gcFax.StatusRemoteID(Device)

```

```

    If gcFax.StatusConnectSpeed(Device) > 0 Then
        frmFaxStatus.lblSpeed = gcFax.StatusConnectSpeed(Device)
    Else
        frmFaxStatus.lblSpeed = ""
    End If

```

```

    If gcFax.StatusPages(Device) Then
        frmFaxStatus.lblPage = CStr(gcFax.StatusPagesSent(Device)) + " of " + Str$(gcFax.StatusPages(Device))
    Else
        frmFaxStatus.lblPage = CStr(gcFax.StatusPagesSent(Device))
    End If

```

```

    If gcFax.StatusPercentage(Device) > 0 Then
        frmFaxStatus.lblPercent = CStr(gcFax.StatusPercentage(Device)) + " % Complete"
    Else
        frmFaxStatus.lblPercent = ""
    End If

```

```

    frmFaxStatus.lblStatus = gcFax.Status(Device)
    frmFaxStatus.lblDestination = gcFax.StatusDestination(Device)
    frmFaxStatus.lblFaxNumber = gcFax.StatusNumber(Device)
End Sub

```

frmMain.frm - MDIForm\_Load

77

**Private Sub MDIForm\_Load()**

```

On Error Resume Next
Me.Left = CLng(GetINISetting(gsAppIniFileSpec, "Windows", "Main Left", "1000"))
Me.Top = CLng(GetINISetting(gsAppIniFileSpec, "Windows", "Main Top", "1000"))
Me.Width = CLng(GetINISetting(gsAppIniFileSpec, "Windows", "Main Width", "6500"))
Me.Height = CLng(GetINISetting(gsAppIniFileSpec, "Windows", "Main Height", "6500"))
Me.WindowState = CLng(GetINISetting(gsAppIniFileSpec, "Windows", "Main WindowState", "0"))
On Error GoTo 0
End Sub

```

**Private Sub MDIForm\_Unload(Cancel As Integer)**

```

Dim r As Integer

r = ValidatePatientDataSaved 'make sure any device data has first been saved
'Save Window positions
If Me.WindowState <> vbMinimized Then
    SaveINISetting gsAppIniFileSpec, "Windows", "Main Left", CStr(Me.Left)
    SaveINISetting gsAppIniFileSpec, "Windows", "Main Top", CStr(Me.Top)
    SaveINISetting gsAppIniFileSpec, "Windows", "Main Width", CStr(Me.Width)
    SaveINISetting gsAppIniFileSpec, "Windows", "Main Height", CStr(Me.Height)
    SaveINISetting gsAppIniFileSpec, "Windows", "Main WindowState", CStr(Me.WindowState)
End If

SaveProgramPreferences
End Sub

```

**Private Sub mnuAccessWebSite\_Click()**

```

'if the form is minimized then set it back to normal
Call LogonToWebSite
If frmBrowser.WindowState = vbMinimized Then
    frmBrowser.WindowState = vbNormal
End If
frmBrowser.ZOrder
End Sub

```

**Private Sub mnuFaxConfigure\_Click()**

```

giLatestOptionsTabSelected = 2 'display the fax tab once the dialog is opened
frmOptions.Show vbModal
End Sub

```

**Private Sub mnuFaxSend\_Click()**

```

frmFaxSend.Show
End Sub

```

**Private Sub mnuFaxViewLogs\_Click()**

```

frmFaxLog.Show
End Sub

```

frmMain.frm - mnuFileProperties\_Click

78

```
Private Sub mnuFileProperties_Click()
    frmOptions.Show vbModal
End Sub
```

```
Private Sub mnuFileSave_Click()
    Dim r As Integer
```

```
    If PAT_DATA.sPatientDataFileName = "" Then
        r = SaveDataToNewFile
    Else
        r = SavePatientData(PAT_DATA.sPatientDataFileName)
    End If
```

```
    If r = False Then
        Beep
        MsgBox "An error occurred while attempting to save the data file. It was not saved.", vbCritical, "File Not Saved"
    End If
End Sub
```

```
Private Sub mnuGenError_Click()
```

```
    MsgBox "This is a temporary test error handler. When you click OK, a synthetic error (Divide by 0) will be generated. The same dialog will be shown when any error is generated. It generates a log file that provides valuable information for the developer. This will be removed from the next build.", vbInformation, "Test Error Handler"
    Error 11
End Sub
```

```
Private Sub mnuHelpDeviceDiag_Click()
```

```
    Dim sMSG, sReply As String
    sMSG = "Performing a device diagnostics test could cause loss of vital device information and should be done only with the assistance of technical support."
    sMSG = sMSG + vbCrLf + "Please contact our technical support department at 1-800-777-7777 for a password and assistance."
```

```
    ' Display message, title, and default value.
    sReply = InputBox(sMSG, "Password Required")
    If LCase$(sReply) = "h2o" Then frmDeviceDiagnostics.Show
End Sub
```

```
Private Sub mnuHelpTips_Click()
```

```
    frmTip.Show
    ' if the form is minimized then set it back to normal
    If frmTip.WindowState = vbMinimized Then frmTip.WindowState = vbNormal
    frmTip.ZOrder
End Sub
```

frmMain.frm - mnuReadDeviceData\_Click

79

---

**Private Sub mnuReadDeviceData\_Click()**

```
    frmReadDeviceData.Show
    'if the form is minimized then set it back to normal
    If frmReadDeviceData.WindowState = vbMinimized Then frmReadDeviceData.WindowState = vbNormal
    frmReadDeviceData.ZOrder
End Sub
```

---

**Private Sub mnuSendDeviceData\_Click()**

```
    frmDeviceInitialize.Show
    'if the form is minimized then set it back to normal
    If frmDeviceInitialize.WindowState = vbMinimized Then frmDeviceInitialize.WindowState = vbNormal
    frmDeviceInitialize.ZOrder
End Sub
```

---

**Private Sub mnuViewAllPatients\_Click()**

```
    frmAllPatients.Show
    'if the form is minimized then set it back to normal
    If frmAllPatients.WindowState = vbMinimized Then frmAllPatients.WindowState = vbNormal
    frmAllPatients.ZOrder
End Sub
```

---

**Private Sub mnuHelpAbout\_Click()**

```
    frmAbout.Show vbModal, Me
End Sub
```

---

**Private Sub mnuViewCalendar\_Click()**

```
    frmDosingCalendar.Show
    'if the form is minimized then set it back to normal
    If frmDosingCalendar.WindowState = vbMinimized Then frmDosingCalendar.WindowState = vbNormal
    frmDosingCalendar.ZOrder
End Sub
```

---

**Private Sub mnuViewExplorer\_Click()**

```
    mnuViewExplorer.Checked = Not mnuViewExplorer.Checked      'toggle the state of the check box
    SSListBar1.Visible = mnuViewExplorer.Checked
End Sub
```

---

**Private Sub mnuViewOptions\_Click()**

```
    frmOptions.Show vbModal, Me
End Sub
```

---

frmMain.frm - mnuViewPatientDosingReport

80

**Private Sub mnuViewPatientDosingReport\_Click()**

```

    frmPatientDosingReport.Show
    'if the form is minimized then set it back to normal
    If frmPatientDosingReport.WindowState = vbMinimized Then frmPatientDosingReport.WindowState = vbNormal
    frmPatientDosingReport.ZOrder
End Sub

```

**Private Sub mnuViewPatientSummary\_Click()**

```

    frmPatientSummary.Show
    'if the form is minimized then set it back to normal
    If frmPatientSummary.WindowState = vbMinimized Then frmPatientSummary.WindowState = vbNormal
    frmPatientSummary.ZOrder
End Sub

```

**Private Sub mnuViewStatusBar\_Click()**

```

    If mnuViewStatusBar.Checked Then
        sbStatusBar.Visible = False
        mnuViewStatusBar.Checked = False
    Else
        sbStatusBar.Visible = True
        mnuViewStatusBar.Checked = True
    End If
End Sub

```

**Private Sub mnuViewToolBar\_Click()**

```

    If mnuViewToolBar.Checked Then
        tbToolBar.Visible = False
        mnuViewToolBar.Checked = False
    Else
        tbToolBar.Visible = True
        mnuViewToolBar.Checked = True
    End If
End Sub

```

**Private Sub SSListBar1\_ListItemClick(ByVal ItemClicked As Listbar.SSListItem)**

```

    Select Case SSListBar1.CurrentGroupKey
        Case "Patient Data" 'patient data
            Select Case ItemClicked.Key
                Case "Event Calendar" 'calendar
                    mnuViewCalendar_Click
            Case "Summary" 'summary
                mnuViewPatientSummary_Click
            Case "Dosing Information" 'grid
                mnuViewPatientDosingReport_Click
            Case "All Patients" 'all patients
                mnuViewAllPatients_Click
    End Select

```

```

    Case "Device" 'device data
        Select Case ItemClicked.Key

```

frmMain.frm - SSListBar1\_ListItemClick

81

```

Case "Retrieve Data"      'read device data
    mnuReadDeviceData_Click

Case "Program Device"    'send to DosPro Device
    mnuSendDeviceData_Click

End Select
End Select
End Sub

```

Private Sub tbToolBar\_ButtonClick(ByVal Button As ComctlLib.Button)

Select Case Button.Key

Case "Open"  
mnuFileOpen\_Click

Case "Save"  
mnuFileSave\_Click

Case "Print"  
mnuFilePrint\_Click

Case "Cut"  
mnuEditCut\_Click

Case "Copy"  
mnuEditCopy\_Click

```

Clipboard.Clear
If TypeOf ActiveForm.ActiveControl Is TextBox Then
    Select Case Index
        Case 0 ' Cut.
            Copy selected text to Clipboard.

```

```

Clipboard.SetText ActiveForm.ActiveControl.SelText
Delete selected text.
ActiveForm.ActiveControl.SelText = ""
Case 1 ' Copy.
    Copy selected text to Clipboard.

Clipboard.SetText ActiveForm.ActiveControl.SelText
Case 2 ' Paste.
    Put Clipboard text in text box.
    ActiveForm.ActiveControl.SelText = Clipboard.GetText()
Case 3 ' Delete.
    Delete selected text.
    ActiveForm.ActiveControl.SelText = ""

```